175 PTAS
mi computer
83
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
Penman
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VII-Fascículo 83

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 148508
Impreso en España-Printed in Spain-Agosto 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

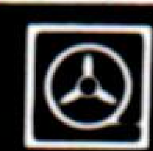
c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.





Piedras en el camino

La introducción de la informática en la educación obliga a los maestros a enfrentarse a un sinnúmero de problemas

La introducción de los ordenadores en la educación se está llevando a cabo con algunas dificultades en la mayor parte de los países. Los recortes económicos, la ignorancia generalizada sobre el tema y las dificultades para educar tanto a los futuros maestros como a los maestros en activo, se combinan para imponer severas restricciones en la informática educativa. La nueva tecnología está sufriendo las recientes políticas gubernativas, y los drásticos recortes presupuestarios resultantes para el sector público están creando una escasez de estos recursos en las escuelas. Tras haber adquirido un costoso hardware, muchos maestros están molestos por los presupuestos escolares que no permiten la adquisición de software para poner las máquinas en funcionamiento.

Asimismo, esta falta de financiación ha impedido la puesta en marcha de importantes proyectos de investigación dedicados a la informática y el niño. Se ha prestado muy poca atención a la forma en que influyen los ordenadores en los programas de estudio vigentes y, por consiguiente, no se ha desarrollado una estrategia clara para introducirlos en las escuelas. Tanto el TICCIT y el PLATO, en Estados Unidos, como el Programa Nacional de Desarrollo, en Gran Bretaña, se llevaron a cabo en los años del "pre-micro", a comienzos de los setenta, cuando la financiación podía obtenerse con relativa facilidad. Ahora, en los años ochenta, fiscalmente tan duros, los gobiernos de ambos países no pueden disponer de los recursos para financiar proyectos similares que son urgentemente necesarios. Entre otros problemas, se incluye el hecho de que no se hayan llevado a cabo estudios serios sobre el lugar que ocupan los ordenadores en el plan de estudios. En las escuelas secundarias, la importancia concedida al BASIC y la existencia de la asignatura de informática como disciplina académica por derecho propio están siendo cuestionadas. A las escuelas primarias se les entrega un ordenador y se les dice que "lo hagan servir", y los maestros deben salir del paso a duras penas, de la mejor manera posible.

En ocasiones, dentro del mundo de la informática educativa se producen graves fallos de comunicación. En Londres, muchas escuelas han invertido en máquinas RML y, por consiguiente, el ILECC (Inner London Educational Computing Centre) invirtió muchísimo tiempo y esfuerzos tratando de escribir una versión de LOGO para los micros RML. Mientras tanto, la propia Research Machines Limited producía una implementación completa y estimable del lenguaje. El ILECC, malgastando dinero para intentar recuperar las pérdidas, se dedicó entonces a producir su propio programa de gráficos de tortuga, menos ambicioso, y ahora se halla ante

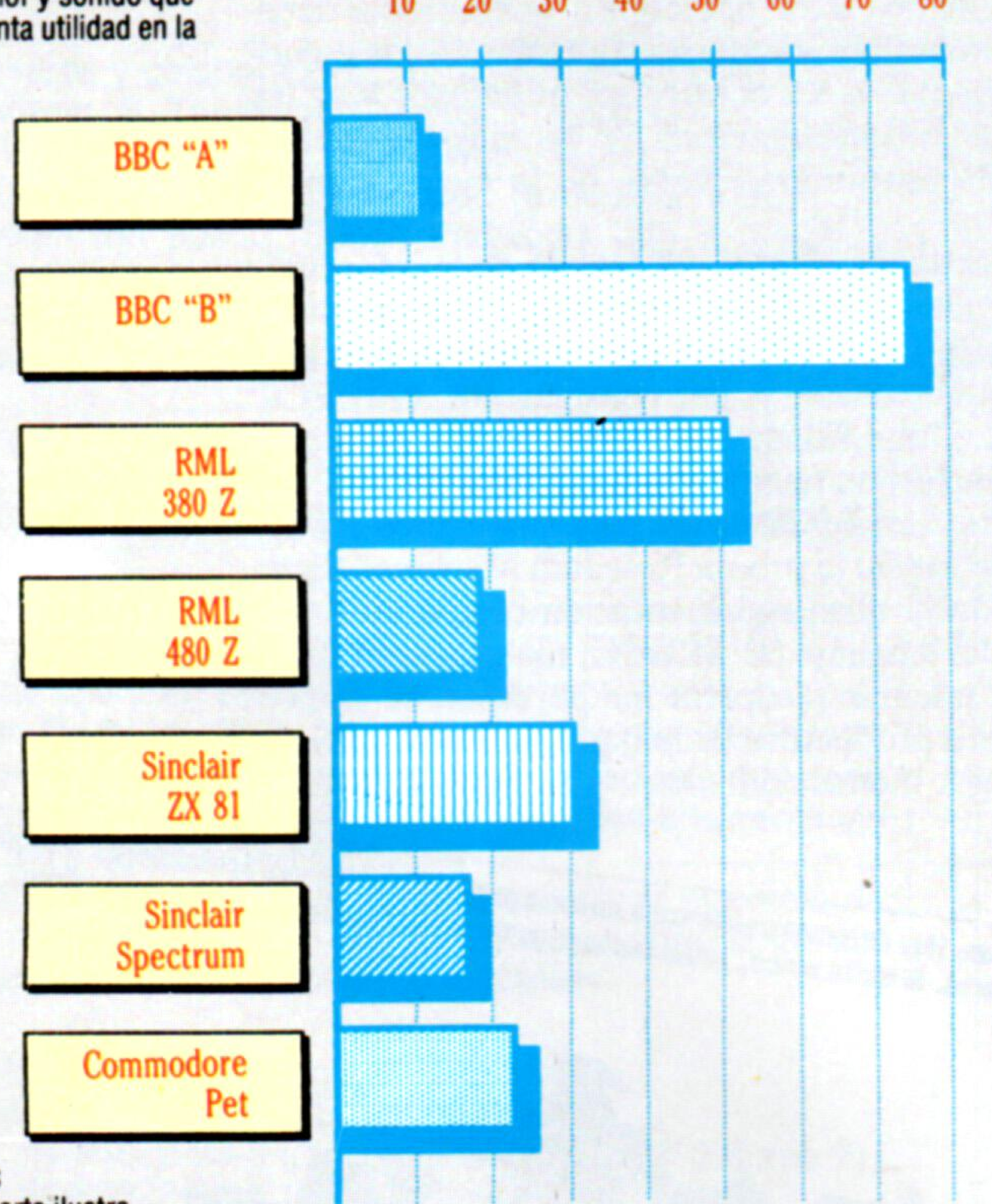
"Buenos días, niños. Hoy daremos comienzo a nuestra primera lección con ordenadores. Tengo aquí siete ordenadores. Si quito cinco, ¿cuántos ordenadores me quedan? A ver tú, Juan, dime..."

Los favoritos de la clase
Respaldadas por la BBC y recomendadas por el gobierno británico para su uso en las escuelas, las máquinas BBC de Acorn se han asegurado una posición sólida en el sistema educativo del país. Las cifras reflejadas en estas páginas están tomadas de un informe encargado por el gobierno que publicó en enero de 1985 la unidad de investigación de servicios de emisiones educativas de la BBC. Una característica sorprendente de los resultados es la continuada presencia de la antigua serie Pet de Commodore y del Sinclair ZX81, que poseen en ambos casos versiones limitadas de BASIC y carecen de las facilidades de color y sonido que pueden ser de tanta utilidad en la clase

la difícil situación de tener que promocionarlo en las escuelas cuando en el mercado ya existe un producto claramente superior.

Se han planteado otros problemas a consecuencia de la ausencia de una política definida para la formación de los maestros. Se han colocado los ordenadores en las clases, esperando que los maestros aprendieran todo a partir del manual. Las restricciones financieras inhibieron la realización de cursos para maestros en activo y, cuando se dispusieron de los mismos, las escuelas se mostraron reacias a enviar a sus maestros debido a la desorganización que se producía en la escuela debido a su ausencia.

La situación en las escuelas de profesores británicas es aún peor. En 1983, el Departamento de Educación estaba preocupado por la falta de cursos



Tipos de micros en escuelas

Tarta de micros
Este gráfico de tarta ilustra claramente que a pesar de los intentos del gobierno británico por mejorar la proporción alumnos/micro en las escuelas del país, ¡aún hoy, más de una escuela de cada cinco tiene que arreglarse con menos de un ordenador por cada 200 niños!

sobre ordenadores para los aspirantes a maestro. Con el apoyo del Departamento de Industria aconsejaron vivamente que cada curso constara de al menos 20 horas dedicadas al papel de los ordenadores en la educación. Chris Gregory, catedrático de Matemáticas en el Bradford College y miembro del MEP (Microcomputers in Education Project, proyecto relacionado con la función del microordenador en la educación), ha afirmado: "Las 20 horas representan un gesto simbólico patético y, aun así, a muchos colegios les resulta difícil implementar siquiera esa cantidad." Un problema acuciante es hallar personas experimentadas. El Departamento de Educación está congelando o recortando el número de personal docente y, en esta situación de reducciones y conflictos con los sindicatos, es difícil reclutar personal con experiencia en ordenadores.

Los programas educativos de calidad han tardado mucho en aparecer. Los gobiernos han invertido en hardware pero no en software y, dado que las escuelas no tienen dinero para gastar, el mercado de software educativo está muy deprimido. En la feria anual de editores de software educativo de 1983 quedó claro que la mayoría de las empresas de software dedicadas a este campo estaban perdiendo dinero y se esperaba que muchas de ellas interrumpieran su actividad durante el año siguiente. Lo que contribuye a deprimir aún más el mercado para los editores de software educativo es el hecho de que muchas de las autoridades más importantes en materia de educación han invertido dinero en desarrollar su propio software, que se distribuye a bajo coste, o gratuitamente, en las escuelas. Esto, como es lógico, hace que los programas comerciales más caros pierdan parte de su atractivo.

Por ejemplo, Sloane Software ha lanzado recientemente un programa de matemáticas denominado *Hopslide*, para niños de entre cinco y siete años. Es un programa sencillo, diseñado para generar una comprensión del valor y el orden de los números. Al niño se le presenta una fila de números entre cinco y nueve. El niño hace saltar los números unos sobre otros o los desplaza una posición. El precio de *Hopslide* es modesto y puede que llegue a venderse bastante bien. La Inner London Education Authority también produce su propio software. El nombre Hopslide también aparece en un programa

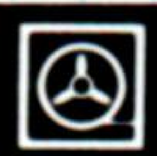
22 %
1 micro por 200 o más
14 %
1 micro por 50
29 %
1 micro por 51-100
20 %
1 micro por 101-150
15 %
1 micro por 151-200

Prop. alumnos/micro

Ian McKinnell

de un paquete de matemáticas producido por su centro de matemáticas, SMILE. Se trata del mismo juego que la versión de Sloane Software, con la excepción de que la gama de números es del dos al nueve y que los gráficos son ligeramente mejores. El paquete SMILE contiene otros 19 programas de similar calidad y vale apenas un poco más que el *Hopslide* de Sloane Software.

La falta de financiación y la deficiente organización no se limitan, por supuesto, a la informática educativa; pero existen algunos otros problemas, menos evidentes, que se plantean cuando se intenta introducir la nueva tecnología en las escuelas. La presencia de los ordenadores, por ejemplo, ha causado problemas. En muchas escuelas, equipos elec-



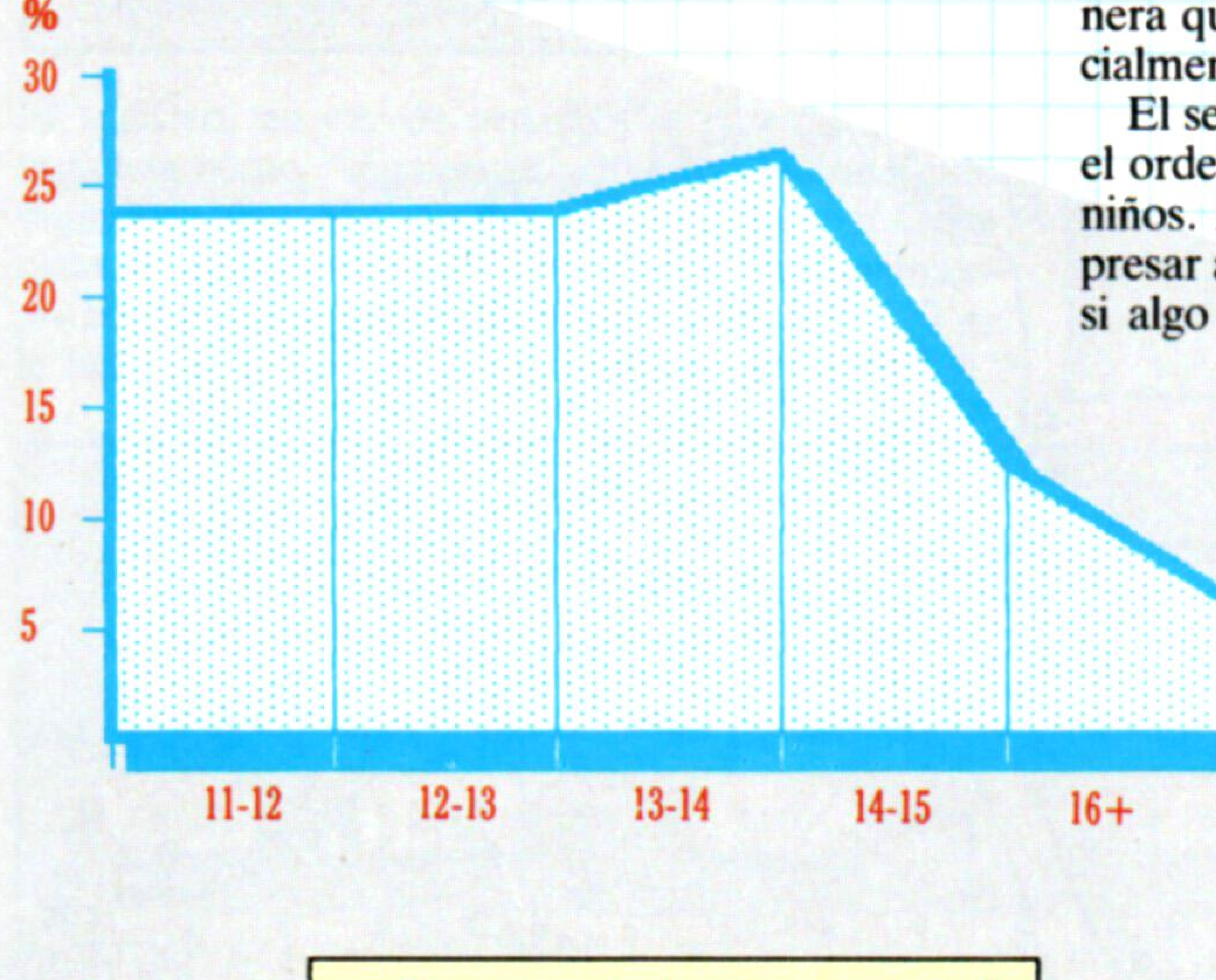
trónicos se almacenan durante la noche en una bóveda de seguridad. Antes de la llegada de los ordenadores, el equipo estaba compuesto por dos o tres cassettes y aparatos de video. En una escuela primaria con un micro por clase, los maestros habían de hacer dos o tres viajes con un ordenador, pantalla, cables, unidad de disco y software a lo largo de un extenso corredor y subir dos plantas por las escaleras, por la mañana y por la tarde. Tras dos accidentes, los maestros se negaron a transportar el equipo y las autoridades educativas correspondientes se negaron a permitir que los ordenadores se dejaran en la clase durante la noche. El resultado fue que todas las máquinas quedaron durante cuatro meses en la bóveda de seguridad, mientras el sindicato de maestros y las autoridades educativas negociaban un acuerdo. Finalmente, el conflicto se resolvió instalando cerca de cada aula armarios de acero para almacenamiento.

En muchas escuelas la situación en materia de seguridad es deplorable, por la sencilla razón de que las autoridades educativas no poseen dinero para invertir en sistemas de seguridad razonables. La mayoría de las escuelas de los barrios centrales superpoblados de las ciudades de Gran Bretaña son objeto de robo o vandalismo varias veces al año. Las pólizas de seguro son tan costosas que a muchas autoridades les resulta más barato reemplazar los bienes sustraídos que asegurarlos.

Entre otros problemas podemos citar el de la discriminación sexual, especialmente porque tanto en la escuela como en el hogar son los varones los que se sienten más atraídos por el ordenador. Los usuarios de la mayoría de los ordenadores personales son varones adolescentes que tienden a utilizarlos para practicar juegos que con frecuencia son violentos y de corte militarista. Esta actitud se ha reproducido en la clase: a menudo el ordenador se considera como un objeto reservado a los varones. Muchos programas educativos están orientados hacia el sexo masculino y, a menudo, los varones monopolizan el ordenador a costa de las mujeres. Un portavoz de la Unidad de Igualdad de Oportunidades de la Inner London Education Authority manifestó: "Somos conscientes de la predilección por el varón en las clases de informática. En las escuelas sólo para mujeres éstas cursan la asignatura con toda satisfacción. El problema sólo se plantea en las escuelas mixtas, donde los niños tienden a dominar. En las escuelas mixtas, pensamos que debemos estimular activamente a las niñas para que se interesen por los ordenadores y restablezcan el equilibrio." Uno de los resultados interesantes que se obtuvieron en la investigación realizada en la Universidad de Edimburgo, utilizando el LOGO en las escuelas, fue que atrajo a las niñas en la misma medida que a los niños.

Otra dificultad importante es que los gobiernos continúan considerando la educación en términos de productividad y costo. Considerar el ordenador como una forma de mejorar la productividad educativa tiene unas implicaciones atemorizadoras para el futuro, con los niños convirtiéndose en parte de una línea de producción informatizada, los maestros reemplazados por máquinas y los planes de estudio y las lecciones determinadas por comisiones del gobierno. Como señaló Seymour Papert: "Existen en el mundo fuerzas formidables que empujan hacia una sociedad más rígida, hacia el empleo del ordenador de una forma que haga la vida menos atractiva, más rígida y menos social. Esas fuerzas son poderosas. ¿Qué podemos hacer frente a ellas?... Podemos decir: 'Quememos los ordenadores.' Eso es una pérdida de tiempo..., o podemos tratar de empujar con la mayor fuerza posible en la dirección contraria y usar el ordenador de una manera que nos permita avanzar en una dirección socialmente deseable."

El sector de la sociedad que finalmente decide si el ordenador es objeto de uso o abuso es el de los niños. Éstos nunca experimentan timidez para expresar a viva voz sus opiniones y son conscientes de si algo es beneficioso y estimulante, o aburrido e irrelevante. Reconocen el software bien pensado y responden a él, y a menudo es tan intensa la atracción que sienten hacia la máquina, que es difícil apartarlos de ésta. En un entorno rico en ordenadores, los niños crean su propia "cultura informática", desarrollando su propia jerga, ayudándose entre sí para resolver los problemas de programación, compartiendo con los demás sus descubrimientos, intercambiando software, educándose a sí mismos colectivamente y evaluando el hardware y el software. Serán los niños, y no los maestros ni los consejeros ni las comisiones ni los investigadores, quienes en última instancia decidan de qué forma se pueden utilizar los ordenadores en la educación para que rindan los mejores frutos.



¿Quién falta?
Este gráfico, que representa la experiencia "sobre teclado" de ordenador entre los escolares de Gran Bretaña, muestra los efectos de especialización en el extremo superior de la gama de edades, donde los estudiantes mayores pueden optar por asignaturas que no exijan uso de ordenadores. Nótese que ninguno de los grupos de edades goza de un porcentaje de "experiencia sobre teclado" que supere el 30 %

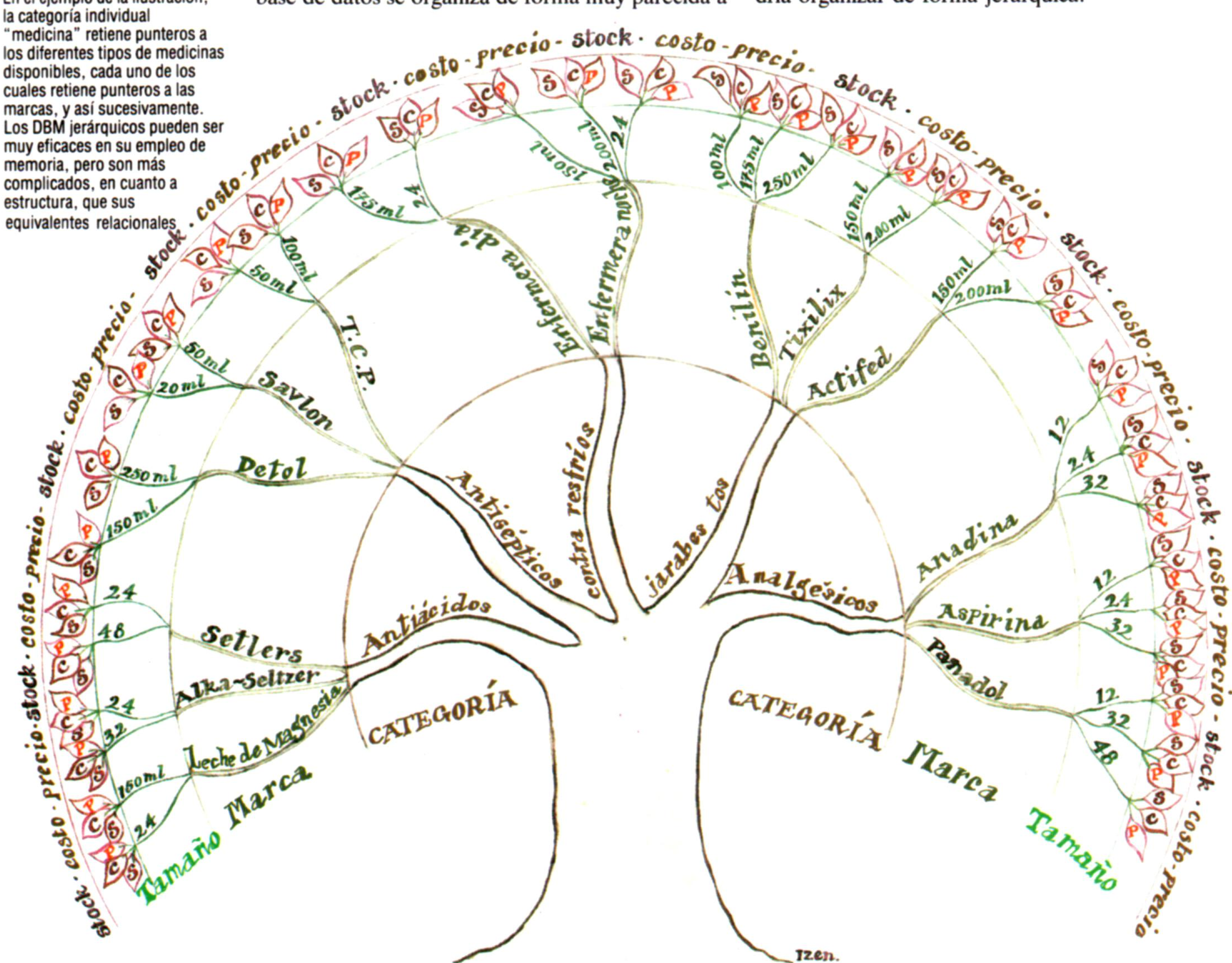
Buenas relaciones

Analizaremos las dos formas más comunes de estructurar una base de datos: los sistemas jerárquicos y en red

La mayor parte de los DBM disponibles en ordenadores personales son del tipo conocido como sistemas *relacionales* de administración de bases de datos. En esencia, esto significa que cada registro de un archivo DBM toma la forma de una tabla, compuesta por filas y columnas, al estilo de una hoja electrónica. Las diversas formas en las que se puede presentar la información sugieren una estructura más compleja, pero ésta, básicamente, no es nada más que una sencilla tabla o cuadrícula. La base de datos se organiza de forma muy parecida a la que emplearía usted para pasar la información en papel. Cada registro es idéntico estructuralmente y cada campo del registro tiene una longitud fija.

El término *relacional* proviene del hecho de que cada "fila" de la base de datos está claramente relacionada, de una forma fija y rígida, a cada "columna". Ésta no es una forma muy flexible de organizar los datos, pero sí sirve para programas administradores de bases de datos relativamente sencillos. Debido a que los ordenadores personales (incluso las nuevas máquinas de 16 bits con 128 K de memoria o más) poseen comparativamente memorias pequeñas, bajas velocidades de proceso y limitadas capacidades para almacenamiento de datos, las restricciones de un sistema relacional forman parte del precio que se debe pagar por combinar una informática al alcance del presupuesto con una administración de bases de datos.

Un enfoque completamente diferente a esta organización tabular de los datos consiste en ordenarlos de forma jerárquica. Podemos pensar que los datos se organizan como si fueran un árbol, cada rama con sus propias bifurcaciones, éstas con sus pequeñas ramitas, e incluso hojas. Para ilustrar esto, crearemos una base de datos del stock de una tienda. En primer lugar, la presentaremos de forma relacional, y después le mostraremos cómo se podría organizar de forma jerárquica.

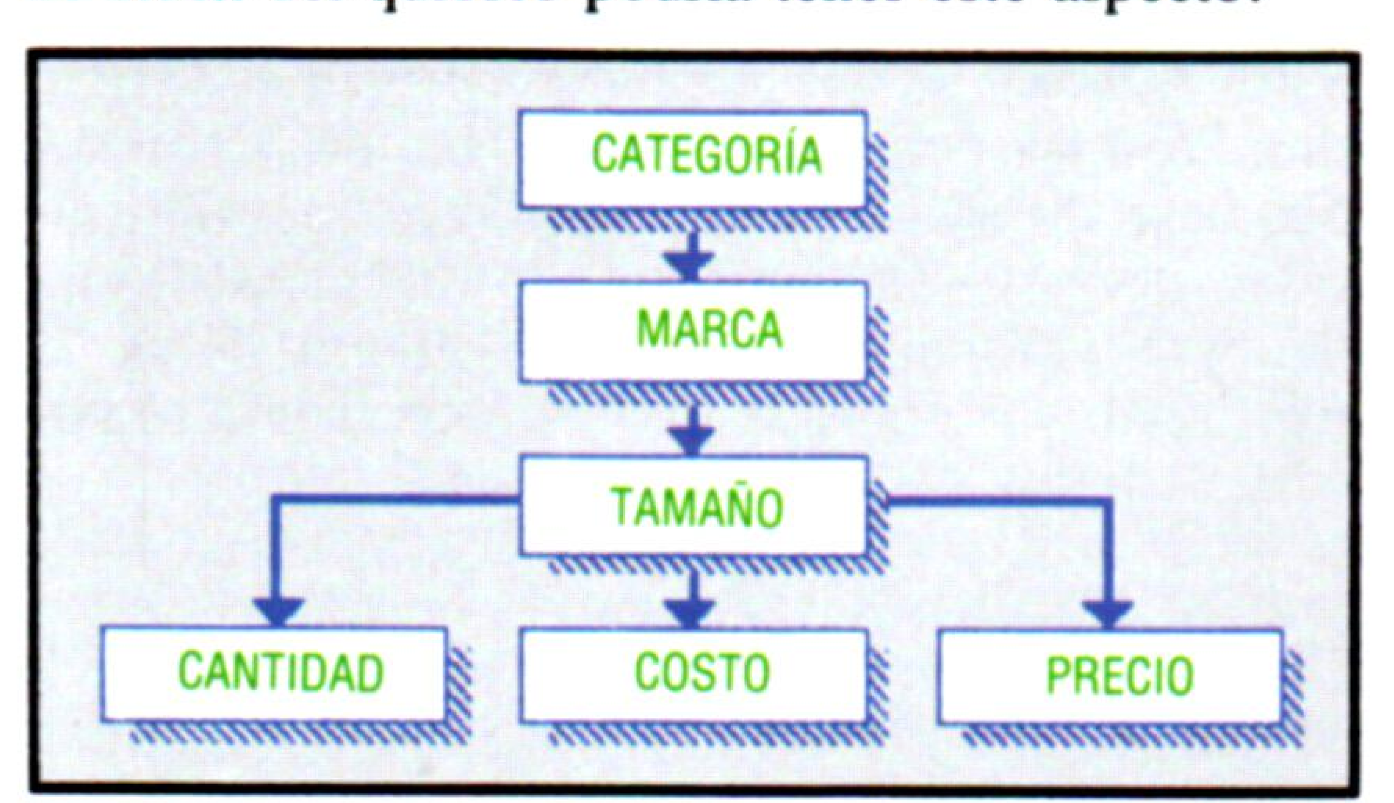


Árbol medicinal
Mientras que un DBM "relacional" almacena la información de forma tabular, un DBM "jerárquico" organiza sus datos en forma de "árbol" En el ejemplo de la ilustración, la categoría individual "medicina" retiene punteros a los diferentes tipos de medicinas disponibles, cada uno de los cuales retiene punteros a las marcas, y así sucesivamente. Los DBM jerárquicos pueden ser muy eficaces en su empleo de memoria, pero son más complicados, en cuanto a estructura, que sus equivalentes relacionales

Gabrielle Izen

CATEGORÍA	MARCA	TAMAÑO	STOCK	COSTO	PVP
CIGARRILLO	EMBASSY	20	350	98	105
CIGARRILLO	EMBASSY	10	140	50	56
CIGARRILLO	DUNHILL INT	20	85	108	125
CIGARRILLO	SILK CUT MILD	20	105	106	126
CARAMELOS	BOUNTY	1	106	14	17
CARAMELOS	TWIX	1	95	12	16
CARAMELOS	MARATHON	1	25	15	19
REVISTA	YOUR SPECTRUM	1	35	85	95
REVISTA	NEW STATESMAN	1	12	60	80
REVISTA	CITY LIMITS	1	86	50	60
BEBIDAS	LUCOZADE	150	35	25	37
BEBIDAS	TIZER	150	40	18	27
BEBIDAS	QUOSM	150	20	16	25

Representada jerárquicamente, esta base de datos de stock del quiosco podría tener este aspecto:



El registro, en vez de organizarse por campos, se organiza como "segmentos" de datos, donde cada segmento puede contener más de un campo. Organizadas jerárquicamente en vez de relacionalmente, las entradas correspondientes a CIGARRILLO en la base de datos relacional tendrían este aspecto:

CIGARRILLO
EMBASSY
DUNHILL INT
SILK CUT MILD
de 10
de 20
de 20
de 20
140 50 56
350 98 105
85 108 125
105 106 126

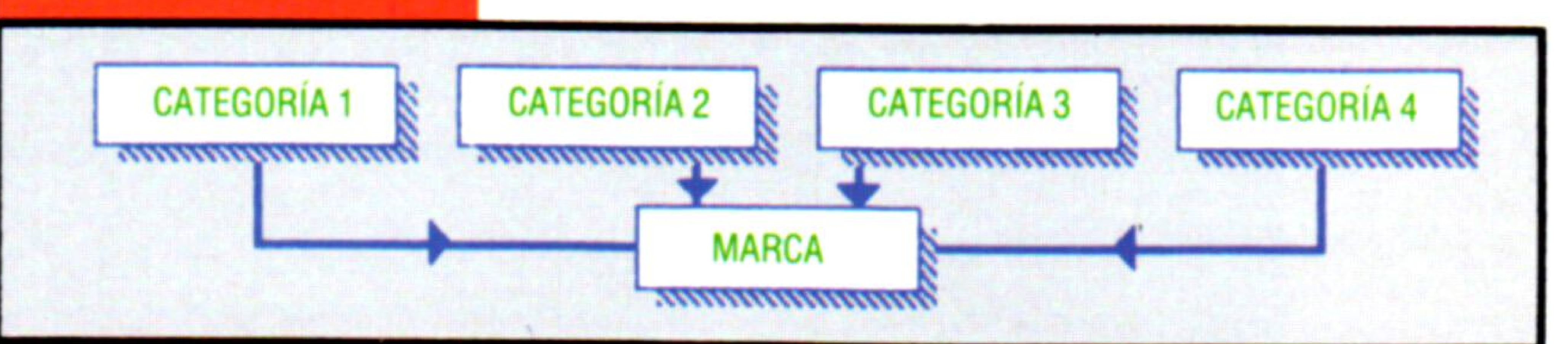
Liz Dixon

Representado de esta forma, cada elemento de la información puede verse claramente como superior o subordinado a otro elemento de ésta. Todo cuanto se requiere es un "puntero" de un segmento de datos a otro. En este ejemplo sólo necesitamos un segmento CIGARRILLO, con punteros a los diversos tipos de cigarrillos que hay en el stock, y otros punteros a tamaño, costo, PVP, etc.

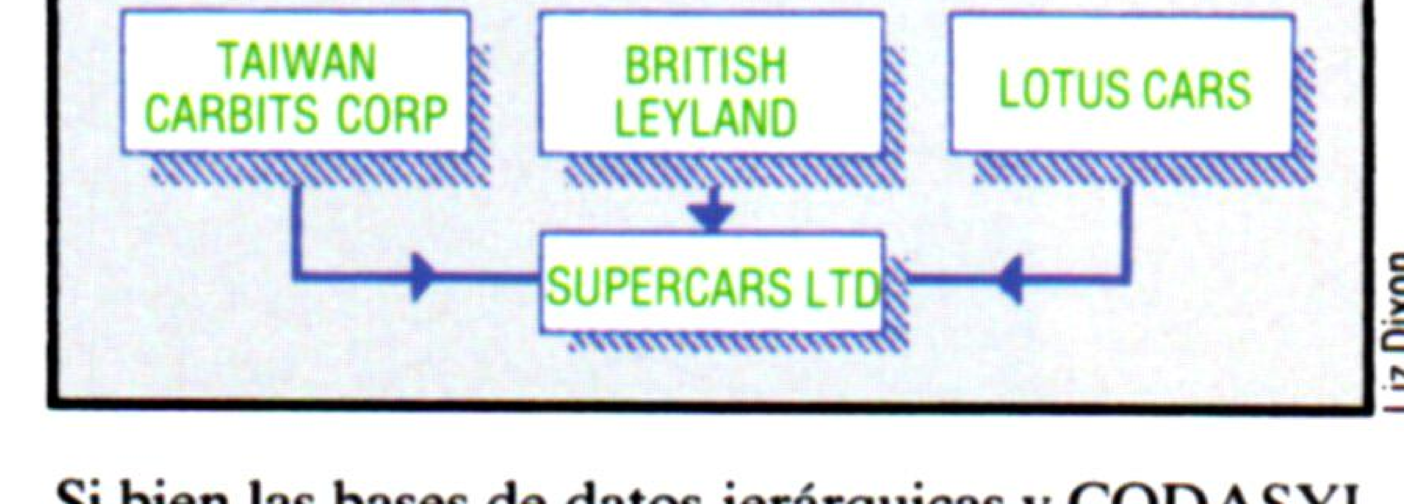
Supongamos que el tendero tiene un centenar de "categorías" en stock, pero mil líneas en venta; con una base de datos relacional, requeriría mil registros: uno para cada línea. Con una base de datos jerárquica, para cubrir todo el stock sólo se necesitaría un centenar de segmentos. Ahorra una gran cantidad de duplicación, pero hace que la base de datos sea más complicada estructuralmente.

Existe aún otro tipo de organización de bases de datos, que se conoce como CODASYL o sistema de red, una organización estandarizada de bases de datos que especificó el grupo de trabajo de bases de datos del Congreso sobre Lenguajes para Sistemas de Datos, CODASYL (Conference on Data Systems Languages). El problema de una base de datos jerárquica es que los datos sólo se pueden organizar para que fluyan en una "dirección"; para utilizar la analogía del árbol, una rama no puede estar unida a dos troncos. Pero, retomando el ejemplo de nuestro tendero, EMBASSY PANATELLAS, por ejemplo, podría ser una "rama" tanto de EMBASSY como de CIGARROS. El problema se plantea particularmente cuando un "componente" simple posee más de un "fabricante". Si usted vende recambios para automóviles, la misma "junta de enlace del extremo mayor" para un BL Marina la podría fabricar la British Leyland, por ejemplo. Como podemos ver, la relación entre mercancías suministradas y proveedores de componentes se puede volver muy complicada.

En el sistema CODASYL se utiliza una red de conjuntos en la que cada conjunto se compone de una colección de registros. La longitud del registro no ha de ser fijada con anterioridad, y cualquier registro puede pertenecer a más de un conjunto. El sistema CODASYL permite que un conjunto esté compuesto por un solo registro, y si hubiera varios conjuntos del mismo tipo, un registro no podría pertenecer a más de uno de ese tipo. Por tanto, esta estructura no está permitida:



Esto representa una limitación seria. Aunque podríamos desear no tener CIGARRILLOs, REVISTAs y BEBIDAs de una misma marca (FABULAN, p. ej.), es fácil imaginar que una empresa llamada Supercars Ltd. formara parte de una estructura como:



Liz Dixon

Si bien las bases de datos jerárquicas y CODASYL son más flexibles que las relacionales, son más complejas y realmente necesitan la potencia de miniordenadores o de ordenadores centrales. Los ordenadores personales utilizan, casi invariablemente, sistemas relacionales. Esta limitación se evidencia en aquellas situaciones en que, por ejemplo, usted tiene un inventario de componentes (libros u otras mercancías) que vende, y cada componente (libro, etc.) posee más de un proveedor. Por lo tanto, necesita un registro de componentes y un conjunto de registros para proveedores relacionados para cada componente. Las DBM de archivos múltiples permiten abrir más de un archivo a la vez y que el archivo activo se remita a un archivo subsidiario.

Puertas abiertas

Veamos cómo trabajan los dos chips CIA 6526 que controlan la comunicación con el exterior en el Commodore 64, analizando el programa "Parawedge"

El Commodore 64 tiene dos chips 6526 CIA (Adaptador de interface complejo), encargados de la comunicación con el mundo exterior. No son ellos los únicos chips con capacidades de E/S; tanto el chip 6510 como el de video manejan también aspectos de E/S. Un chip 6526 tiene dos puertas de datos de ocho bits, ambas con líneas programables por separado. El chip soporta la comunicación de 16 bits o de 8 bits y tiene dos temporizadores de 16 bits enlazables. Posee además un registro de desplazamiento de 8 bits para la comunicación en serie, y, como ya vimos, un reloj de hora-del-día de 24 horas programable.

Tiene además dos líneas específicas de enlace (*handshaking*): la PC y la Flag. La PC bajará durante un ciclo después de que los datos sean escritos en la puerta B del 6526, y puede emplearse para indicar a un dispositivo externo que están preparados los datos (*data ready*). La línea Flag puede usarse como una entrada de control desde otro dispositivo.

Esto puede servir para poner a uno el bit Flag en el registro de interrupción y, si es el caso, originar una interrupción no enmascarable (NMI) al procesador 6510.

En el Commodore 64 los dos chips 6526 tratan aspectos diversos de las E/S: uno de ellos (CIA#1, con la dirección de base en $DC00) se destina al teclado y a las palancas de mando, mientras que el otro (CIA#2, con dirección de base en $DD00) controla los datos en las puertas seriales y del usuario. El chip video maneja las E/S al monitor, y la puerta de cassette es tratada directamente por el 6510.

El programa Parawedge que damos aquí ilustra con claridad los pasos necesarios para programar el 6526 directamente para E/S. La rutina es una cuña NMI que emplea la línea Flag. Está diseñada para enviar un bloque específico de la memoria por la puerta para el usuario como datos paralelos, o bien para recibir datos paralelos de 8 bits hasta que el bloque especificado de la memoria esté lleno. Dado que se trata de una codificación de cuña, temporizará los datos que entran o salen sobre NMIs, dejando la máquina libre para que realice otras tareas en el tiempo restante. El único inconveniente es que si la velocidad de transmisión de datos es demasiado alta el Commodore 64 comenzará a gastar todo el tiempo ejecutando rutinas de servicio NMI, lo que puede ser muy engorroso.

El programa Parawedge permite que el Commodore 64 establezca comunicaciones a dos vías de 8 bits en paralelo con un dispositivo externo (otro ordenador, una impresora en paralelo) a través de la puerta para el usuario. Las patillas de la puerta para el usuario desde PB0 hasta PB7 se emplean para el traslado de datos; la patilla Flag 2 se emplea como entrada de enlace; la PA2 da la señal de *ready for data* (preparado para datos), y la PC2 da la señal *data valid* (dato válido). Para emplear el programa, se ha de definir primero un área de RAM desde la cual se desea enviar los datos que salen, o sobre la cual se desea recibir los datos entrantes. Esto se consigue pasando al programa las direcciones de inicio y de final (en la forma *lo-hi*) colocándolas (POKE) en las cuatro posiciones que van de la 50768 a la 50771.

La posición 50772 se emplea para indicar si el programa va a realizar una operación de Entrada o Salida de datos. Si se coloca (POKE) un uno en esta posición el programa quedará preparado para salida; si se coloca un cero, el programa estará en modo entrada. Una vez establecidos estos parámetros, la codificación de cuña se inicia por medio de SYS 50775. Dado que el programa está conducido por interrupciones, se ejecutará entre bastidores, enviando o recibiendo datos, mientras se digita o ejecuta un programa cualquiera en BASIC en primer plano.

Entrada del Parawedge
Incluimos aquí el listado en código assembly del Parawedge, que puede digitarse y ensamblarse por medio de un ensamblador. El programa puede, igualmente, ser introducido como una serie de sentencias DATA, digitando y ejecutando la codificación del cargador del BASIC

Programa Parawedge para el Commodore 64

Cargador del BASIC

```
1000 REM ** CARGADOR DEL BASIC DEL PARAWEDGE **
1010 DATA173,84,198,208,61,169,0,141,3
1020 DATA221,169,144,141,13,221,173,2
1030 DATA221,9,4,141,2,221,173,0,221,9
1040 DATA4,141,0,221,173,80,198,133,251
1050 DATA173,81,198,133,252,173,24,3
1060 DATA141,85,198,173,25,3,141,86,198
1070 DATA120,169,188,141,24,3,169,198
1080 DATA141,25,3,88,96,169,255,141,3
1090 DATA221,169,144,141,13,221,173,24
1100 DATA3,141,85,198,173,25,3,141,86
1110 DATA198,120,169,234,141,24,3,169
1120 DATA198,141,25,3,88,96,169,144,44
1130 DATA13,221,240,36,173,1,221,145
1140 DATA251,230,251,208,2,230,252,173
1150 DATA82,198,197,251,173,83,198,229
1160 DATA252,144,49,173,0,221,41,252
1170 DATA141,0,221,9,4,141,0,221,108,85
1180 DATA198,169,144,44,13,221,240,246
1190 DATA177,251,141,1,221,230,251,208
1200 DATA2,230,252,173,82,198,197,251
1210 DATA173,83,198,229,252,144,3,108
1220 DATA85,198,120,173,85,198,141,24,3
1230 DATA173,86,198,141,25,3,88,108,24
1240 DATA3
1250 DATA25596:REM*SUMA DE CONTROL*
1260 CC=0
1270 FOR I=50775TO50971
1280 READX:CC=CC+X:POKE I,X
1290 NEXT
1300 READX:IFCC<>XTHENPRINT"ERROR SUMA DE
     CONTROL"
1310 END
```



Listado en assembly

```
;  ************************************************
;  *                                              *
;  *  PARAWEDGE - PROGRAMA CUÑA DE ENVIO/RECEPCION *
;  *                                              *
;  *       EN COMUNICACIONES DE 8 BITS            *
;  *                                              *
;  *     EN PARALELO EN EL COMMODORE 64           *
;  *                                              *
;  *                                              *
;  ************************************************
;
   CIA 2=$DD00          ; DIRECCION DE BASE DEL CHIP 6526
   OUTPUT=$FF
   INPUT=$00
   OUTSHK=$04
   INTMSK=$90
   TOGHI=$04
   TOGLO=$FC
   NMIVEC=$0318
   ZPTEMP=$FB
   *=$C650
   START *=*+2          ; DIRECCION DE INICIO
   END *=*+2            ; DIRECCION DEL FINAL
   MODE *=*+1           ; FLAG DE E/S
   VECTOR *=*+2         ; ALMACEN PARA EL VECTOR NMI

   LDA  MODE            ;ENTRADA O SALIDA
   BNE  OUTDAT          ;BIFURCACION SI ES SALIDA
   LDA  #INPUT
   STA  CIA2+3          ;ESTABLECE DDR PARA ENTRADA
   LDA  # INTMSK
   STA  CIA2+13         ;DESACTIVA LAS INTERRUPCIONES DEL FLAG
   LDA  CIA2+2
   ORA  #OUTSHK
   STA  CIA2+2          ;PONE PA2 PARA SALIDA
   LDA  CIA2
   ORA  #TOGHI
   STA  CIA2            ;PONE ALTA LA LINEA PA2 DE ENLACE
   LDA  START
   STA  ZPTEMP
   LDA  START+1         ;MUEVE PUNTEROS A PAGINA CERO
   STA  ZPTEMP+1
;
;INICIALIZACION CUÑA ENTRADA
;
   LDA  NMIVEC
   STA  VECTOR          ;GUARDA EL ANTIGUO VECTOR NMI
   LDA  NMIVEC+1
   STA  VECTOR+1
   SEI                  ;
   LDA  #<NXTIN         ;
   STA  NMIVEC          ;INSERTA LA CUÑA DE ENTRADA DATOS
   LDA  #>NXTIN         ;
   STA  NMIVEC+1        ;
   CLI                  ;
   RTS
;
;INICIALIZACION CUÑA SALIDA
;
OUTDAT
   LDA  #OUTPUT
   STA  CIA2+3          ;PONE DDR PARA SALIDA
   LDA  #INTMSK
   STA  CIA2+13         ;DESACTIVA INTERRUPCIONES FLAG
;
;
   LDA  NMIVEC
   STA  VECTOR
   LDA  NMIVEC+1        ;GUARDA ANTIGUO VECTOR NMI
   STA  VECTOR+1
   SEI                  ;
   LDA  #<NXTOUT        ;
   STA  NMIVEC          ;INSERTA LA CUÑA DE SALIDA DATOS
   LDA  #>NXTOUT        ;
   STA  NMIVEC+1        ;
   CLI                  ;
   RTS
;
;
;RUTINA DE SERVICIO DATOS ENTRADA
;
;
NXTIN
   LDA  #INTMSK         ;COMPRUEBA ICR
   BIT  CIA2+13         ;INTERR DEBIDA A FLAG?
   BEQ  NOTCOM          ;NO... NMI NORMAL
;
;BYTE OK EN PUERTA
;
   LDA  CIA2+1          ;LEE EL BYTE
   STA  (ZPTEMP),Y      ;LO ALMACENA EN MEMORIA
   INC  ZPTEMP
   BNE  TEST1           ;INCREMENTA EL PUNTERO
   INC  ZPTEMP+1
;
TEST1
   LDA  END
   CMP  ZPTEMP
   LDA  END+1           ;COMPRUEBA SI ES EL FINAL
   SBC  ZPTEMP+1
   BCC  DONE            ;BIFURCACION SI SE HA ACABADO
;
;AVISA A DISPOSITIVO PREPARADO PARA BYTE SIGUIENTE
;
   LDA  CIA2
   AND  #TOGLO
   STA  CIA2            ;PONE PA2 BAJA Y DESPUES ALTA
   ORA  #TOGHI
   STA  CIA2
;
;REALIZA AHORA LA RUTINA NORMAL NMI
;
NOTCOM
   JMP (VECTOR)
;
;
;RUTINA DE SERVICIO DATOS SALIDA
;
;
NXTOUT
   LDA  #INTMSK         ;COMPRUEBA ICR
   BIT  CIA2+13         ;INTERR. DEBIDA AL FLAG?
   BEQ  NOTCOM          ;NO... EFECTUA NMI NORMAL
;
;ENVIO OK BYTE
;
   LDA  (ZPTEMP),Y      ;TOMA BYTE DE LA MEMORIA
   STA  CIA2+1          ;LO SACA.EL PC BAJARA
                        ;DURANTE UN CICLO
   INC  ZPTEMP
   BNE  TEST2           ;INCREMENTA EL PUNTERO
   INC  ZPTEMP+1

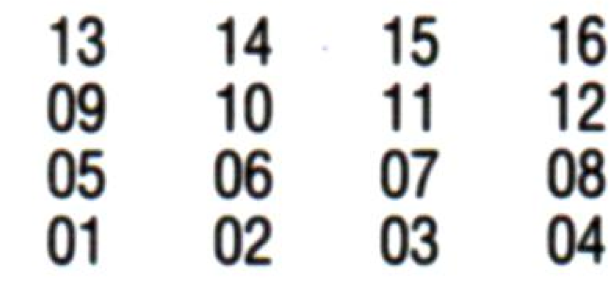
TEST2
   LDA  END
   CMP  ZPTEMP
   LDA  END+1           ;COMPRUEBA SI ES EL FINAL
   SBC  ZPTEMP+1
   BCC  DONE            ;BIFURCACION SI SE HA ACABADO
;
;CONTINUA RUTINA NMI NORMAL
;
   JMP (VECTOR)
;
;
;TERMINADO QUITAR CUÑA
;
;
DONE
   SE I
   LDA  VECTOR
   STA  NMIVEC          ;RESTAURA EL VECTOR NMI A SU
   LDA  VECTOR+1        ;VALOR INICIAL
   STA  NMIVEC+1
   CL I
   JMP  (NMIVEC)
```

Parawedge está tomado de *Mastering the Commodore*, de Jones y Carpenter, y se imprime por cortesía de los autores y de la Ellis Horwood Ltd.

Cuatro en raya

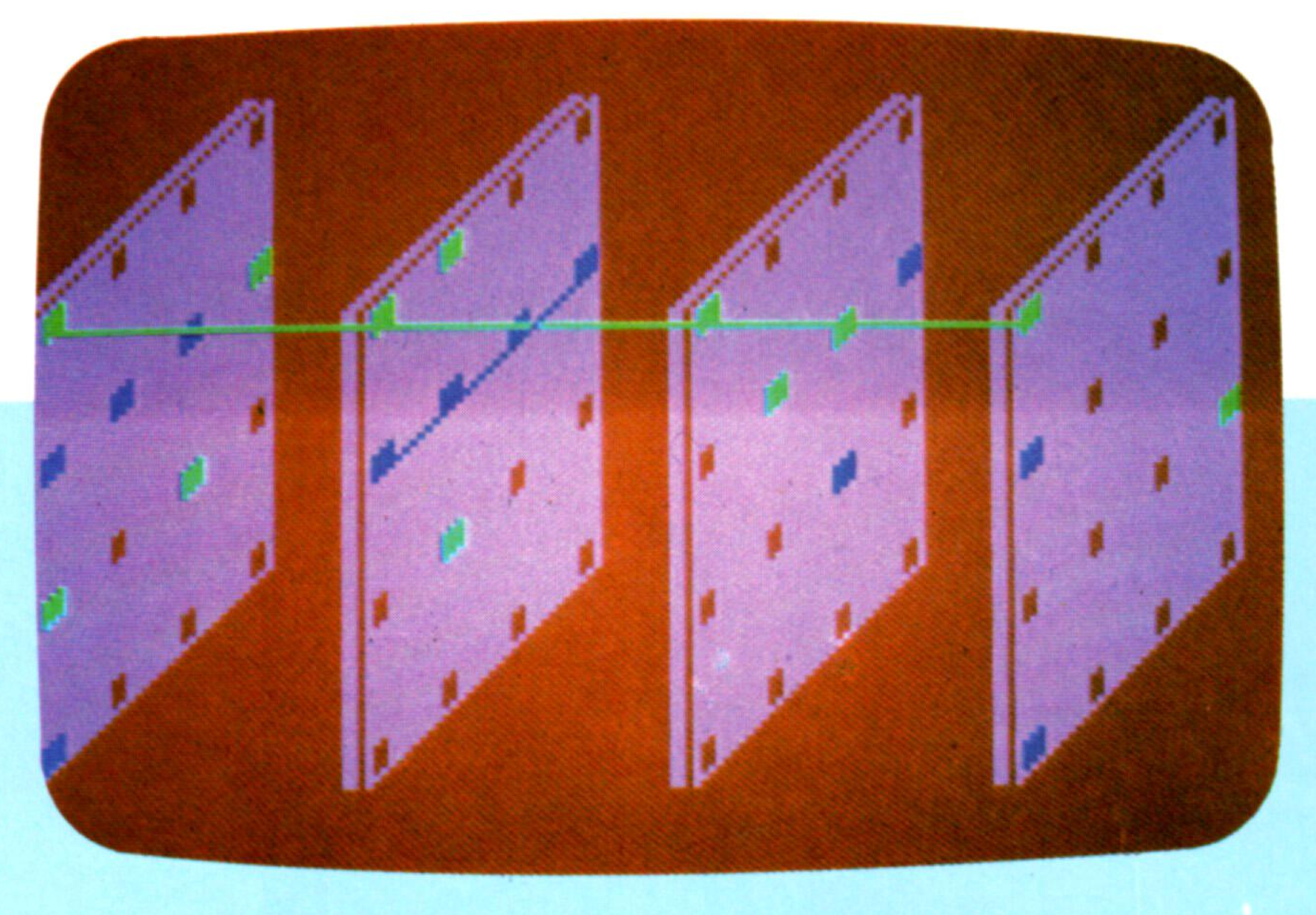
Este juego en tres dimensiones exige 24 K de memoria. Le permite jugar contra el ordenador Atari u otro adversario de su elección

Para ganar hay que colocar cuatro cuadrados de igual color en línea. Usted juega en 4 planos (A, B, C, D), divididos en 16 cuadrados. Para jugar, primero debe elegir un plano y luego digitar un número correspondiente al cuadrado escogido. Éstos están numerados así:



13	14	15	16
09	10	11	12
05	06	07	08
01	02	03	04

He aquí ejemplos de pulsaciones (recuerde que es inútil pulsar la tecla "Return"): D15, A05, B10, B16, C02. Cuando el ordenador le solicite el número de jugadores, digite 1 para jugar contra la máquina o 2 para enfrentar a otro adversario.



```
0 REM ***************************************
1 REM *          CUATRO EN RAYA             *
2 REM *        EN TRES DIMENSIONES          *
3 REM ***************************************
4 GOSUB 3000
5 DIM SPACE(64,2):DIM MARK(64):DIM BLOB(
  86,3):DIM HZ(40)
6 CL=0
7 GOSUB 6000
10 GRAPHICS 7+16:COLOR 1
17 SETCOLOR 2,12,8:SETCOLOR 4.0.3
18 SETCOLOR 1,4,8:SETCOLOR 0,2,6:SETCOLO
   R 3.3.8
20 FOR X=0 TO 129 STEP 43
30 FOR Y=95 TO 30 STEP -1
40 PLOT X,Y:DRAWTO X+30,Y-30
50 NEXT Y
60 NEXT X
61 PLOT 0,28:DRAWTO 28,0:PLOT 0,27:DRAWT
   O 27.0
62 FOR RT=0 TO 86 STEP 43
63 PLOT RT+41,95:DRAWTO RT+41,30:DRAWTO
   RT+71,0:PLOT RT+40,95:DRAWTO RT+40,30:DR
   AWTO RT+70.0
64 NEXT RT
65 COLOR 0:SETCOLOR 4,9,2
66 FOR Z=0 TO 129 STEP 43
70 FOR Y=91 TO 31 STEP -20
72 FOR X=1 TO 31 STEP 9
74 FOR A=0 TO 1
760 PLOT Z+X+A,Y-A-X+3:DRAWTO Z+X+A,Y-A-X
780 NEXT A
850 NEXT X
900 NEXT Y
950 NEXT Z
1000 FOR P=1 TO 64:READ H,J
1010 SPACE(P,1)=H:SPACE(P,2)=J
1012 NEXT P
1015 IF PEEK(764)=255 THEN 1015
1016 CLOSE #1:OPEN #1,4,0,"K:":GET #1,H
1017 IF W<65 OR H>68 THEN POKE 764,255:6
     OTO 1015
1018 IF W=65 THEN N=0:IF W=66 THEN N=16:
     IF W=67 THEN N=32
1019 IF W=66 THEN N=16
1020 IF W=67 THEN N=32
1021 IF W=68 THEN N=48
1023 POKE 764,255
1024 IF PEEK(764)=255 THEN 1024
1025 CLOSE #5:OPEN #5,4,0,"K:":GET #5,Z
1026 IF Z<48 OR Z>49 THEN POKE 764,255:6
     OTO 1024
1027 POKE 764,255
1028 POKE #4:OPEN #4,4,0"K:":GET #4,HH
     :IF WH<48 OR WH>57 THEN POKE 764,255:GOT
     O 1027
1033 REM FOR TR=0 TO 2
1034 SP=N+10*(Z-48)+WH-48:LOCATE
     SPACE(SP,1),SPACE(SP,2),GH:IF GH<>0 THEN
     GOTO 1015
1035 CL=CL+1:IF CL=2 THEN CL=0
1036 FOR RY=15 TO 0 STEP -1:COLOR (RY+CL
     +2):SETCOLOR 1,3,4
1037 FOR TR=0 TO 2
1038 PLOT SPACE(SP,1)+TR,SPACE(SP,2)-TR:
     DRAWTO SPACE(SP,1)+TR,SPACE(SP,2)-TR-3
1039 NEXT TR:NEXT RY
1040 CLOSE #1
1042 FOR PQ=1 TO 64
1044 LOCATE SPACE(PQ,1),SPACE(PQ,2),V
1046 MARK(PQ)=V:NEXT PQ
1050 C=CL+2
1100 A=1:B=16:D=16:E=1:GOSUB 1400
1105 A=1:B=61:D=1:E=4:GOSUB 1400
1110 A=1:B=13:D=17:E=4:GOSUB 1400
1115 A=4:B=16:D=15:E=4:GOSUB 1400
1120 A=1:B=4:D=20:E=1:GOSUB 1400
1125 A=13:B=16:D=12:E=1:GOSUB 1400
1130 A=1:B=1:D=21:E=1:GOSUB 1400
1135 A=13:B=13:D=13:E=1:GOSUB 1400
1140 A=16:B=16:D=11:E=1:GOSUB 1400
1145 A=4:B=4:D=19:E=1:GOSUB 1400
1150 A=1:B=49:D=5:E=16:GOSUB 1400
1155 A=4:B=52:D=3:E=16:GOSUB 1400
1160 A=1:B=4:D=4:E=1:GOSUB 1400
1165 A=17:B=20:D=4:E=1:GOSUB 1400
1170 A=33:B=36:D=4:E=1:GOSUB 1400
1175 A=33:B=36:D=4:E=1:GOSUB 1400
1180 A=49:B=52:D=4:E=1:GOSUB 1400
1299 IF JOUEUR=1 AND CL<>0 THEN GOTO 150
     0:REM IF CL=0 THEN GOTO 1015
1300 GOTO 1015
1400 FOR PQ=A TO B STEP E
1410 IF MARK(PQ)=C AND MARK(PQ+D)=C AND
     MARK(PQ+2*D)=C AND MARK(PQ+3*D)=C THEN G
     OSUB 1450
1420 NEXT PQ
1430 RETURN
1450 PLOT SPACE(PQ,1)+1,SPACE(PQ,2)-1:DR
     AWTO SPACE(PQ+3*D,1)+1,SPACE(PQ+3*D,2)-1
     :RETURN
1500 REM JUEGO ORDENADOR
1600 FOR PQ=1 TO 64
1605 LOCATE SPACE(PQ,1),SPACE(PQ,2),V
1606 IF V=2 THEN V=10
1608 IF V=3 THEN V=50
1610 MARK(PQ)=V:NEXT PQ
1615 H=0
1620 A=1:B=16:D=16:E=1:GOSUB 1700
1625 A=1:B=61:D=1:E=4:GOSUB 1700
1630 A=1:B=13:D=17:E=4:GOSUB 1700
1635 A=4:B=16:D=15:E=4:GOSUB 1700
1640 A=1:B=4:D=20:E=1:GOSUB 1700
1645 A=13:B=16:D=12:E=1:GOSUB 1700
1650 A=1:B=1:D=21:E=1:GOSUB 1700
1655 A=13:B=13:D=13:E=1:GOSUB 1700
1660 A=16:B=16:D=11:E=1:GOSUB 1700
1665 A=4:B=4:D=19:E=1:GOSUB 1700
1670 A=1:B=49:D=5:E=16:GOSUB 1700
1675 A=4:B=52:D=3:E=16:GOSUB 1700
1680 A=1:B=4:D=4:E=1:GOSUB 1700
1682 A=17:B=20:D=4:E=1:GOSUB 1700
1686 A=33:B=36:D=4:E=1:GOSUB 1700
1688 A=49:B=52:D=4:E=1:GOSUB 1700
1690 GOTO 1740
1700 FOR PQ=A TO B STEP E
1705 H=H+1
1710 SUM=MARK(PQ)+MARK(PQ+D)+MARK(PQ+2*D
     )+MARK(PQ+3*D)
1720 IF SUM=30 THEN GOTO 1830
1725 BLOB(H,1)=SUM:BLOB(H,2)=BLOB(H,3)=D
1730 NEXT PQ:RETURN
1740 FOR TZ=1 TO H:IF BLOB(TZ,1)=150 THE
     N GOTO 1850
1745 NEXT TZ
1746 YZ=0:FOR TZ=1 TO H
1748 IF BLOB(TZ,1)=20 OR BLOB(TZ,1)=100
     THEN GOSUB 1860
1750 NEXT TZ:IF YZ<>0 THEN GOTO 1760
1755 GOTO 2050
1760 FOR TZ=1 TO YZ:FOR PZ=2 TO (YZ-1)
1765 IF HZ(TZ)=HZ(PZ) AND TZ<>PZ THEN SP
     =HZ(TZ):GOTO 1035
1766 NEXT PZ:NEXT TZ
1770 IF YZ<>0 THEN SP=HZ(YZ):GOTO 1035
1825 GOTO 2050
1830 FOR TK=0 TO 3:IF MARK(PQ+TK*D)=0 TH
     EN SP=PQ+TK*D:GOTO 1035
1835 NEXT TK
1850 REM DEFENSA
1855 FOR TK=0 TO 3:IF MARK(BLOB(TZ,2)+TK
     *BLOB(TZ,3)=0 THEN SP=BLOB(TZ,2)+TK*BLO
     B(TZ,3):GOTO 1035
1857 NET TK
1860 FOR P=0 TO 3
1863 IF MARK(BLOB(TZ,2)+P*BOB(TZ,3))=0
     THEN YZ=YZ+1:HZ(YZ)=BLOB(TZ,2)+P*BLOB(TZ
     ,3)
1865 NEXT P:RETURN
1890 GOTO 2050
1950 GOTO 2050
2050 SP=INT(64*RND(1)+1):IF MARK(SP)=0 T
     HEN GOTO 1035
2055 GOTO 2050
3000 REM PANTALLA ENTERA
3020 GRAPHICS 2+16:SETCOLOR 4,13,2
3030 POSITION 3,1:?#6;" #################### "
3040 POSITION 3,2:?#6;" #       CUATRO      #"
3050 POSITION 3,3:? #6;"#       EN RAYA     #"
3052 POSITION 3,4:? #6;"#       EN TRES     #"
3054 POSITION 3,5:? #6;"#     DIMENSIONES   #"
3060 POSITION 3,6:? #6;" #################### "
3070 POSITION 1,7:? #6;"      PULSE START      "
3080 IF PEEK(53279)=6 THEN RETURN
3090 GOTO 3080
5000 DATA 1,93,10,84,19,75,28,66
5010 DATA 1,73,10,64,19,55,28,46
5020 DATA 1,53,10,44,19,35,28,26
5030 DATA 1,33,10,24,19,15,28,6
5040 DATA 44,93,53,84,62,75,71,66
5050 DATA 44,73,53,64,62,55,71,46
5060 DATA 44,53,53,44,62,35,71,26
5070 DATA 44,33,53,24,62,15,71,6
5080 DATA 87,93,96,84,105,75,114,66
5090 DATA 87,73,96,64,105,55,114,46
5100 DATA 87,53,96,44,105,35,114,26
5110 DATA 87,33,96,24,105,15,114,6
5120 DATA 130,93,139,84,148,75,157,66
5130 DATA 130,73,139,64,148,55,157,46
5140 DATA 130,53,139,44,148,35,157,26
5150 DATA 130,33,139,24,148,15,157,6
6000 GRAPHICS 2+16:? #6;"  2 jugadores O"
6010 ? #6;"1 JUGADOR CONTRA MI"
6020 ? #6;"(PULSE 1 O 2)"
6030 OPEN #1,4,0,"K:":GET #1,M:IF M<49 O
     R M>50 THEN POKE 764,255:CLOSE #1:GOTO 6
     030
6040 JOUEUR=M-48:CLOSE #1
6050 RETURN
```



Tres en uno

El Penman Plotter, creado para el BBC Micro, es un dispositivo que puede funcionar como trazador de gráficos, tortuga o "ratón"

La gran cantidad de interfaces existentes en el BBC Micro, así como su amplia aceptación entre los especialistas de la educación, se han combinado para producir una gran cantidad de periféricos de carácter didáctico para la máquina. Aquí ya hemos examinado un gran número de dispositivos que se pueden controlar mediante el BBC Micro, incluyendo máquinas tan diversas como plotters (dispositivos trazadores de gráficos) ratones y robots móviles. No obstante, el Penman Plotter es un periférico que con toda justicia se puede considerar como estos tres dispositivos fusionados en uno.

El paquete Penman Plotter se compone de una unidad de control, un plotter móvil, una fuente de alimentación eléctrica, un cable conector RS232 y el software que lo acompaña. Cuando no se está utilizando, la unidad de control y el plotter se unen entre sí para conformar una única unidad compacta, que mide sólo 55 por 128 por 335 mm. Para separar el plotter de la unidad de control sólo hay que pulsar el clip que hay debajo de la misma, deslizar el plotter hacia afuera y desenrollar el cable plano conector desde el interior mismo de la unidad de control. Una vez extraído de su lugar, el dispositivo deja al descubierto tres agujeros en los que se colocan los lápices, así como una copa situada exactamente en el medio, donde se puede colocar otro lápiz para gráficos de tipo tortuga. Los lápices, de punta de fieltro y 40 mm de longitud, una vez colocados en sus copas, se mantienen sobre el papel mediante clips sustentados por resortes. Cuando se da la orden de dibujar, el resorte es empujado hacia abajo por los electroimanes situados en las barras de dentro, y el peso del lápiz presiona la punta contra el papel.

En la cara inferior del dispositivo hay tres ruedas, dos de las cuales son propulsoras; la tercera es una pequeña rueda plástica de movimiento libre que se encarga del equilibrio. Las ruedas propulsoras son de metal y están rodeadas por una cubierta gruesa para proporcionar una fricción adicional cuando el plotter se desplaza sobre el papel. En frente de cada una de ellas hay un sensor luminoso, que detecta el margen del papel registrando la diferencia de brillo entre el papel y el material sobre el cual está colocado.

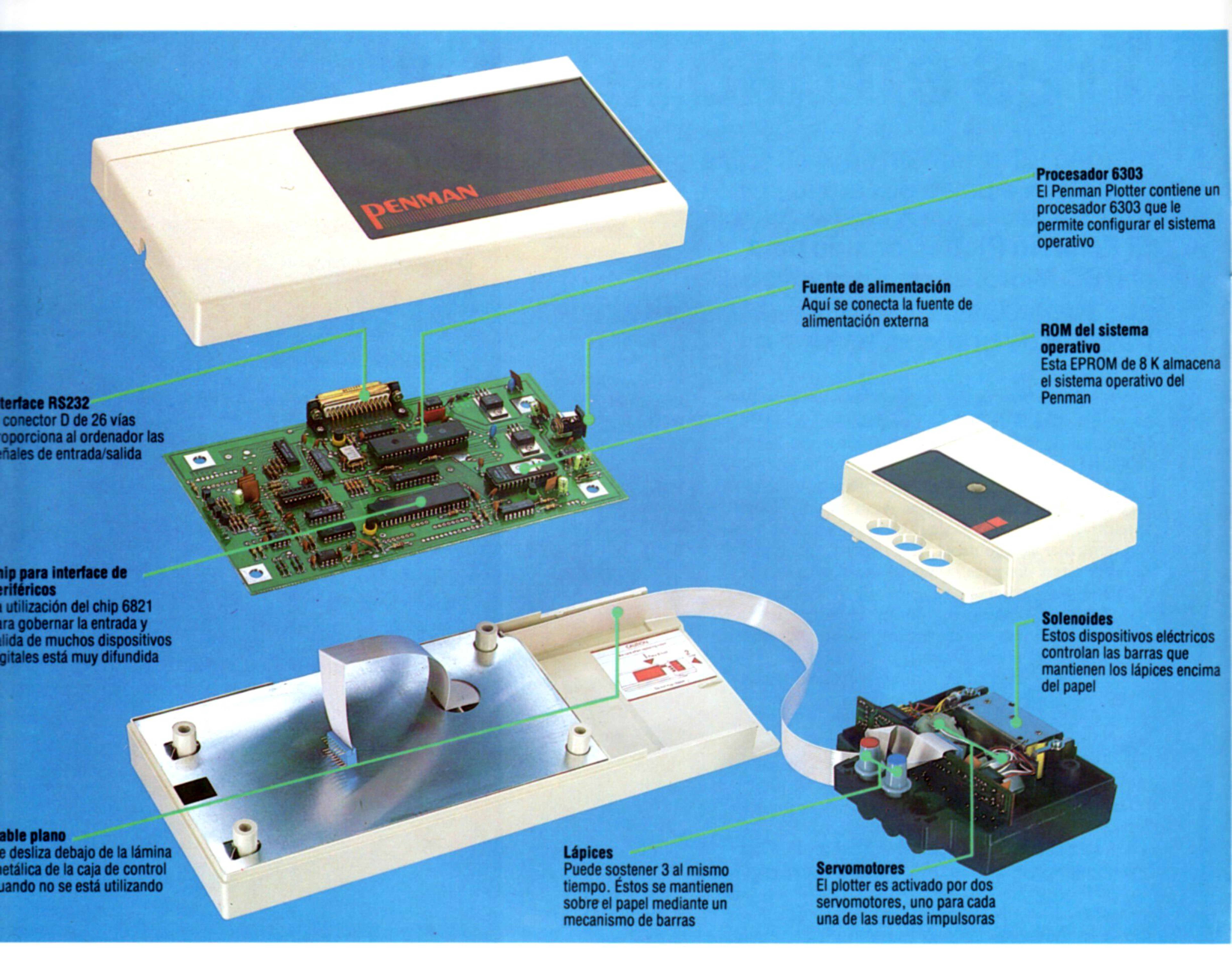
Dentro del dispositivo trazador de gráficos hay tres electroimanes conectados a barras, cada uno de los cuales controla la elevación y el descenso de un lápiz sobre el papel; junto a ellos se halla el par de motores eléctricos estándares unidos a las ruedas impulsoras. El plotter utiliza motores eléctricos comunes en lugar de motores paso a paso o servo; éstos permiten que el dispositivo trace curvas continuas en vez de curvas "paso a paso". Sin embargo, ello exige que el software de control sea paralelamente más sofisticado, porque ha de ser capaz de variar los voltajes aplicados a los motores, en vez de limitarse a enviar una serie de impulsos.

Por último, hay una placa de circuitos que distribuye la potencia y decodifica las señales que llegan desde la unidad de control y que las envía ya sea a los electroimanes o a los motores. En cada uno de los husillos de los motores hay un disco estroboscópico que gira a medida que rota el motor, y a los lados de las caras del disco hay diodos detectores de luz. A medida que el disco estroboscópico gira, en los diodos aparecen impulsos de luz que le informan a la lógica de la placa de circuitos sobre la rapi-



Chris Stevens

Doble personalidad
El Penman Plotter se compone de un módulo de control, conectado a un ordenador a través de una interface RS232, y una tortuga móvil. El cable plano que une la tortuga a la caja de control es bidireccional, lo que significa que los dos dispositivos se pueden comunicar entre sí. Por consiguiente, mientras la caja de control le envía instrucciones a la tortuga para que se desplace a una posición, la tortuga puede transmitir su posición de vuelta al sistema de circuitos de control. Es esta característica del sistema la que permite utilizar el Penman Plotter no sólo como tortuga y plotter, sino tambien como "ratón"



dez a que se está moviendo el motor, de modo que la máquina pueda calcular su posición. Esta lógica es particularmente útil cuando se utiliza el Penman como "ratón", porque el ordenador necesita saber en qué dirección y a qué velocidad están girando las ruedas para poder desplazar el cursor a través de la pantalla.

La placa de circuito impreso del interior de la unidad de control tiene instalados tres chips principales. La placa contiene un microprocesador 6303 (un desarrollo del 6800), que posee la facilidad de configurar toda una variedad de sistemas operativos. El procesador incorpora una pequeña RAM de trabajo que le permite almacenar la posición del plotter. El segundo de los chips principales de la placa es una ROM para interface de periféricos 6821. También incluida en la placa hay una EPROM que contiene los programas de demostración que llevará a cabo el Penman cuando no esté conectado el cable RS232.

El Penman se puede utilizar en varias modalidades diferentes. La modalidad "emulador de terminal" permite controlar el dispositivo directamente desde el teclado. Para entrar en ella, se debe cargar directamente el programa *Pentlk* o, empleando el robot Penman como ratón, cargar el programa desde el menú principal. Una vez cargado el software, las instrucciones se envían a la unidad de control a través de la puerta RS232 en forma de códigos ASCII. Digitando PRINT 'I' se inicializa el Penman. Él recibirá la señal y determinará cuál de las tres velocidades posibles (300, 1 200, 9 600 baudios) se está utilizando. Hecho esto, el Penman se puede colocar en su secuencia de colocación en la posición inicial digitando la instrucción H. El robot realizará una secuencia de movimientos que hará que el dispositivo se sitúe en el rincón inferior izquierdo del papel.

Desde su posición de partida, el Penman intentará hallar la parte inferior de la página mediante el empleo de sus sensores detectores de luz. Una vez hallado el margen, el dispositivo realizará un giro de 90° y llevará a cabo la misma acción a lo largo del lado izquierdo de la página. Como ayuda para asegurar que el contraste entre el papel y el fondo sea suficiente para que el plotter lo pueda detectar, Penman Products ha incluido sabiamente un trozo de papel negro sobre el cual se puede colocar el papel de dibujo. Una vez que el Penman ha "aterrizado" en su base, establece su posición de partida a

50 mm de cada borde del papel (la distancia desde la parte frontal del plotter hasta la copa central).

La aplicación *Pentlk* se puede ejecutar ya sea en modalidad directa (una instrucción cada vez) o bien se pueden encadenar instrucciones entre sí para formar programas que se pueden cargar (LOAD), guardar (SAVE) o ejecutar (RUN). En la modalidad "emulador de terminal" el movimiento es cartesiano, lo que significa que el papel sobre el cual se coloca al robot se divide en una cuadrícula. Cuando al plotter se le dé la instrucción de ir a (500,500), se moverá hacia estas coordenadas en vez de moverse 500 pasos en ambos sentidos. Una hoja de papel tamaño A4 posee un máximo de 2 100 coordenadas en el eje *x* y 2 970 en el eje *y*. Las instrucciones se pueden entrar tanto en modalidad absoluta como en modalidad relativa, según la instrucción MOVE lleve o no el prefijo de una A o una R. Los lápices se seleccionan con instrucciones parecidas a las del LOGO, tales como U para lápiz arriba, D para lápiz abajo y P para lápiz seleccionado.

En modalidad directa, el Penman Plotter también puede producir movimientos para gráficos de tortuga, si bien éstos son algo más complicados que los movimientos cartesianos, porque la longitud de la distancia a recorrer se debe entrar en notación hexadecimal con una $ como prefijo. Esto se debe a que los gráficos de tortuga pasan por alto el software normal que interpreta los movimientos cartesianos e investigan directamente los bits de las direcciones controladoras del interior del ordenador. Sin embargo, ésta no es la única forma en que se pueden realizar gráficos de tortuga con el Penman Plotter. El software de aplicaciones que se suministra contiene programas que permiten controlar el robot mediante LOGO.

Para controlar al plotter en la modalidad robótica se utiliza un sistema que controla directamente el Penman desde el registro de dirección de datos de la puerta para el usuario. Cada bit del registro controla un aspecto diferente de los movimientos del robot: los bits 0 y 1 y los bits 2 y 3 controlan, respectivamente, los motores derecho e izquierdo. Los bits 4 y 5 llevan a cabo las funciones generales de los motores, como encenderlos y apagarlos, mientras que el bit 6 controla el movimiento del lápiz hacia arriba y hacia abajo. El Penman también devuelve información de sí mismo al registro de datos, tal como errores posicionales y si el dispositivo ha detectado o no el borde de la página con sus sensores de luz.

Al Penman Plotter también se le pueden enviar instrucciones de texto. La gama de instrucciones y sus aplicaciones son muy similares a las que se utilizan en las impresoras/plotter existentes para una gran cantidad de máquinas personales. De hecho, la forma de los caracteres impresos de ambas guardan una notable similitud. El tamaño del texto se puede variar entre 1 y 127 mm en altura y se puede imprimir en una de cuatro direcciones. Una útil adición del Penman Plotter, de la cual no disponen las impresoras/plotter tradicionales, es la capacidad para sesgar el texto.

El manual que viene con el plotter facilita una lista completa de las instrucciones disponibles, con algunas explicaciones sobre cómo están implementadas, así como una explicación del hardware. No obstante, quizá sea un poco avanzado para el principiante, dando la impresión de que el Penman está dirigido a quienes ya poseen un conocimiento profundo de los mecanismos de su ordenador.

El Penman Plotter se puede considerar como otro adelanto en el desarrollo de una tortuga/plotter de propósito general. Utilizado correctamente, la resolución del dispositivo se aproxima a la de los plotters más convencionales que se utilizan en los estudios de diseño comerciales. Sin embargo, por el momento el software impide que el dispositivo sea una herramienta de gestión viable. El juego de programas que actualmente se suministra con el plotter está concebido básicamente como una herramienta educativa. Se espera que los usuarios escriban sus propios programas para hacer funcionar el plotter, aprendiendo, por consiguiente, los principios del software para robótica. Por el contrario, un usuario comercial no se suele preocupar especialmente por los detalles más delicados de la programación, sino que exige algo que sea fácil de usar. Como herramienta educativa, el Penman Plotter tiene un gran valor. La gama de modalidades de operación disponibles hacen que la máquina sea una propuesta interesante para las escuelas.

PENMAN PLOTTER

DIMENSIONES

335×128×55 mm

INTERFACES

RS232C, que permite su conexión a cualquier ordenador compatible con RS232

RESOLUCION

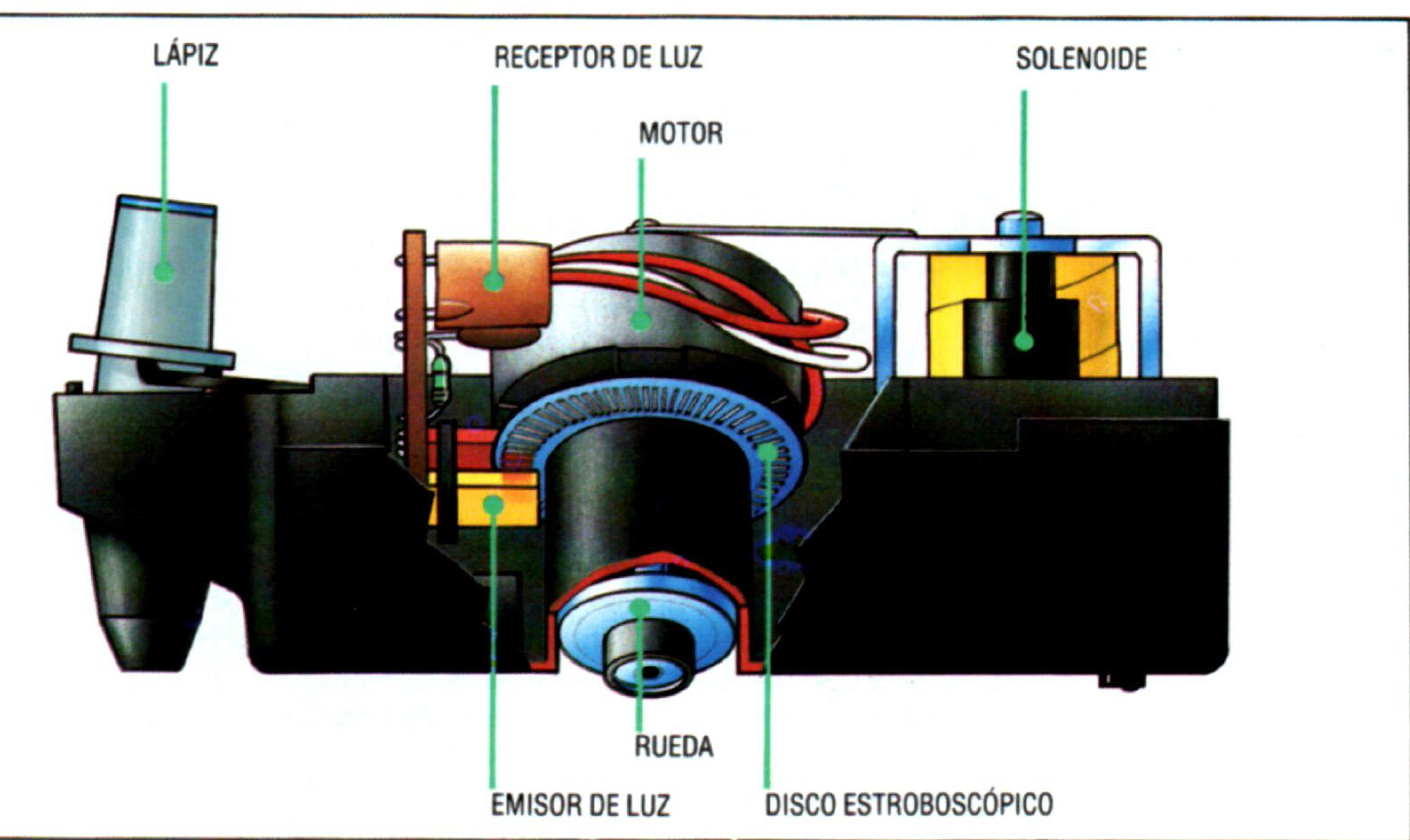
Tamaño del texto: entre 1 y 127 mm; resolución de gráficos: 0,1 mm

DOCUMENTACION

El manual proporciona muchísimos detalles técnicos que permitirán al usuario avanzado desarrollar todo el potencial del Penman, si bien carece de material de formación para el principiante

Al girar la rueda

Para lograr que el sistema de control del Penman "sepa" cuál es la posición del plotter, en el robot debe haber alguna clase de mecanismo de realimentación para contar el número de "pasos" que se han realizado. Para hacer esto, los fabricantes han colocado un delgado disco de metal alrededor de cada uno de los ejes de activación de las ruedas. El disco posee una serie de delgados cortes que van desde el centro hacia el borde. En la placa de circuitos hay montado un dispositivo que emite y recibe luz. A medida que el disco gira, se produce un efecto estroboscópico entre el emisor y el receptor en forma de una serie de impulsos. Se calcula la cantidad de éstos para determinar el número de pasos dado



Kevin Jones

Solos en el mar

Mientras navegamos hacia el Nuevo Mundo, varios eventos incidirán en la duración del viaje y en la salud de los tripulantes

El código para los eventos menores restantes se incluye en una serie de pequeñas secciones del programa que la subrutina de la línea 5500 selecciona al azar. En el capítulo anterior nos ocupamos de los primeros cinco eventos, de modo que cuando añadamos los nuevos habrá un total de 13, lo que exige cambiar el valor de RM, en la línea 46, a 13. También se debe incrementar la lista de números de línea, tras la sentencia ON X GOTO de la línea 5525, para dar cabida a las nuevas rutinas. El programa principal selecciona un evento menor de forma aleatoria, de entre su lista ampliada, para ejecutar cada semana.

```
5525 ON X GOTO 5540,5570,5570,5570,5570,5600,5700,5800,
5850,5900,5950,6000,6050
```

El primer número de línea a añadir a la sentencia ON...GOTO de la línea 5525 es el 5600: aquí comienza la rutina que se ocupa de capturar peces. Si el generador de eventos al azar determina que X sea 6, entonces la línea 5525 enviará el programa a esta rutina. Cuando muere un tripulante, la muerte no se registra en la variable CN hasta el final de la semana; pero si en el transcurso de la semana actual todo el resto de la tripulación hubiera pasado a mejor vida, obviamente no habría nadie para capturar los peces.

El programa comprueba esta eventualidad mediante la creación de un bucle en la línea 5610. Observa la matriz de la tripulación para ver si algún tripulante ha muerto durante la semana, determinando qué tasas de fortaleza se han establecido en −999. Después, la línea 5630 cuenta el número de tripulantes fallecidos y le resta el número al total. Si el resultado es menor que 1, el evento se ignorará y se devolverá el control al programa principal.

Si todavía queda vivo algún tripulante, la línea 5650 genera un número aleatorio entre 11 y 20, representando la cantidad de peces que se han cogido, expresada en kilos. Esta cantidad se suma a la carne restante en la línea 5680 y entonces el programa informa al jugador para cuánto tiempo alcanzarán las provisiones de carne. Por último, en la línea 5685, el total de carne se divide por el número de tripulantes y sus necesidades semanales, produciendo una nueva estimación de la cantidad de semanas para las que alcanzarán las provisiones.

El siguiente evento de la sentencia ON X GOTO es un temporal, que se inicia en la línea 5700. La tripulación se las apaña para recoger el agua de lluvia en barriles y reabastecer sus provisiones de agua. El programa genera un número al azar entre 11 y 20 en la línea 5735 que representa la cantidad de barriles de agua que se han recogido, y que se suma a la cantidad disponible en la línea 5750. Por último, la línea 5755 calcula e imprime la nueva estimación de cuánto tiempo durará el agua.

El octavo evento posible es que soplen vientos favorables, lo que aumentará la velocidad del barco y acortará la duración del viaje en media semana. La línea 5835 resta esta mitad al contador de semanas. La rutina de "buen tiempo" (evento número nueve) comienza en la línea 5850, la novena dirección de la línea 5525. El buen tiempo permite que la tripulación pueda tomarse un descanso para mejorar su estado de salud, lo que se simula aumentando sus tasas de fortaleza en TS(,). En la línea 5882 se prepara un bucle que recorre TS(,). Si se incrementara la tasa de fortaleza de un tripulante fallecido, esa persona "resucitaría", de modo que la línea 5884 comprueba si hay alguna tasa establecida en 0 o −999 (que significa en ambos casos la muerte) y hace caso omiso de ellos. En la línea 5886 se le suma a la tasa de fortaleza de cada tripulante vivo un número al azar entre 5 y 15.

Si el número seleccionado al azar para un evento menor es 10, el programa se dirigirá a la línea 5900, la subrutina de pérdida de medicinas. El mal tiempo ha hecho que se rompieran la mitad de los frascos de medicina, y esto evidentemente tendrá un efecto negativo en el caso de que algún tripulante se ponga enfermo. El programa comprueba primero si queda alguna medicina a bordo, examinando el primer elemento de la matriz de suministros, OA(). Si el valor está establecido en 0 o −999, no

hay ninguna medicina, de modo que este evento se ignorará. De haber alguna medicina a bordo, la línea 5925 divide la cantidad por la mitad; no obstante, puesto que los frascos no se rompen por la mitad, toma del resultado la parte entera. Después el programa imprime un mensaje informando al jugador cuántos frascos quedan.

El undécimo evento se produce cuando una ola inesperada hace que las armas se mojen y se oxiden, quedando inservibles. Las armas son una mercancía importante, que puede necesitarse para la defensa, para cambiar por alimentos o para comerciar. La subrutina que estropea las armas es similar a la que rompe los frascos de medicina. Comprueba si queda algún arma a bordo, en la línea 5955, divide por dos la cantidad en la línea 5975 e imprime la fracción entera del resultado.

El hecho de que los ratones se hayan comido la mitad de las balas de tela representa el evento número doce, que comienza en la línea 6000. La tela es el cuarto elemento de la matriz de suministros, OA(), de modo que el programa comprueba OA(4), en la línea 6005, para ver si queda algo de tela a bordo. Si se ha comprado tela, los ratones, que habitan en la línea 6025, se comen la mitad de ella y, tras el festín, se informa al jugador sobre la cantidad de tela que queda todavía.

Observe que en esta etapa hay una oportunidad de hacer alguna trampa en el juego, haciéndolo más fácil o más difícil mediante la alteración de los factores de los acontecimientos. Los límites entre los números aleatorios que se pueden seleccionar se pueden reducir o ampliar e, igualmente, se pueden aumentar o disminuir las proporciones de los suministros perdidos.

El último de los eventos menores, que, amenazadoramente, es el número 13, es la observación de un albatros. Este evento es único entre los acontecimientos menores, ya que hay dos posibilidades de que se produzca en una misma semana. Si la sentencia ON X GOTO selecciona el evento número 13, el incidente se producirá de la forma normal, como un evento menor. Si el albatros es avistado cuando la tripulación se halla a media ración de carne, se les ofrecerá la oportunidad de abatirlo para conseguir su carne. Los jugadores que conozcan el poema de Coleridge *The rime of the ancient mariner* se sentirán con toda justicia recelosos de hacerlo. En el programa principal, los eventos menores se llaman mediante el GOSUB de la línea 860; los eventos mayores comienzan en la línea 870. En la 875 se produce la segunda posibilidad de avistar el albatros. La línea comprueba si la tripulación se halla a media ración de carne, comprobando si HR(3) es igual a 0.5. De ser así, un factor aleatorio (AND RND(1)<.5) da un 50 % de probabilidades de avistar el ave. Si ésta se avista en estas circunstancias, el programa pasa a la subrutina "albatros", que comienza en la línea 6050.

Avistar un albatros trae buena suerte. En la línea 6055 se establece en "S" una variable, A$, indicando que se ha avistado el ave. Posteriormente, ésta se utilizará en el programa para reducir la posibilidad de un motín, resultado práctico de la buena suerte. El programa verifica luego, en la línea 6075, si la tripulación se está quedando sin carne, determinando si la cantidad de carne restante es igual a, o mayor que, las necesidades semanales de la tripulación multiplicadas por el tiempo que aún falte

Ian McKinnell

En este día, 17 de julio del año de Nuestro Señor de 1589, hemos perdido otro tripulante. El pobre Robert Purkiss, mientras realizaba la segunda guardia, fue arrastrado por la borda. Dios se apiade de su alma.

Su muerte, aunque trágica, al menos ha reducido la cantidad de vituallas necesarias para cada semana. Ahora que la tripulación se halla por debajo de su número óptimo es probable que el viaje dure más tiempo de lo previsto y pienso poner a la tripulación a media ración antes de que avistemos tierra y desembarquemos.

para concluir la travesía. Si las provisiones son suficientes, el ave sigue su rumbo y la línea 6130 retorna al programa principal.

Si las existencias de carne son escasas, se le dice al jugador que el ave pesa 10 kilos y se le pregunta si desea abatirla. La línea 6115 comprueba si, en efecto, desea intentarlo, examinando el primer carácter de la entrada. Si el jugador decide no disparar, el ave sigue su vuelo, pero si éste desea dispararle, el programa comprueba, en la línea 6133, si OA(2) está establecida en 0 (indicando que no hay ningún arma utilizable). De estar establecida en −999, las armas se habrían perdido durante la semana en curso y el programa así le informaría al jugador, mientras el ave se aleja volando.

Si hay armas a bordo, se le dispara el ave. La línea 6140 le da al jugador un 50 % de probabilidades de dar en el blanco, generando un número aleatorio entre 0 y 1. Si el número es mayor que .5 el tiro es errado y el ave se escapa; de lo contrario, se hace blanco y ésta cae en la cubierta. Si durante la semana en curso se hubieran agotado las reservas de carne, el valor correspondiente de la matriz se restablecería en 0, quitando el −999, en la línea 6160, para poder registrar las provisiones extras que proporcionaría el cuerpo del ave. La cantidad de carne de la matriz de provisiones, PA(3), se aumenta en 10 kilos y se le indica al jugador el nuevo total.

El abatimiento del albatros se señala en el poema de Coleridge mediante una interrupción del invitado a la boda:

"¡Dios te guarde, anciano marinero!
de los espíritus malignos que te atormentan.
¿Cuál es la pena que te aflige?"
"Con mi ballesta
abatí al albatros."

En nuestro juego se le da al jugador una advertencia similar. En la línea 6162, B$ se establece en "S". Éste es el factor de mala suerte, que supera con creces a la buena suerte que trajo la línea 6055. La naturaleza exacta de esta mala suerte se revelará más adelante en el proyecto.

Módulo 7: Más eventos aleatorios

Adición al bucle principal del viaje

```
875  IF HR(3)=.5 AND RND(1)<.5 THEN PRINT CHR$(147):GOSUB
     6050
```

Subrutina más eventos al azar

```
5600 REM EVENTO 6 – CAPTURAR PECES
5605 X=0
5610 FOR T=1 TO 16
5615 IF TS(T,2)=–999 THEN X=X+1
5620 REM CONTAR FALLECIDOS ESTA SEMANA
5625 NEXT
5630 IF CN – X<1 THEN RETURN
5635 REM NINGUNA ACCION SI TODA TRIPULACION MUERTA
5640 PRINT
5645 S$="    DURANTE LA SEMANA*":GOSUB 9100
5646 PRINT:GOSUB 9200
5650 S$="UNO DE LOS TRIPULANTES CAPTURO*":GOSUB 9100
5655 X=INT (RND(1)*10)+11
5660 REM ENTRE 10 y 20 KILOS
5662 PRINT X; "KILOS DE PESCADC"
5665 PRINT:GOSUB 9200
5670 S$="AHORA TU PROVISION DE CARNE ES*":GOSUB 9100
5675 S$="SUFICIENTE PARA APROXIMADAMENTE*":GOSUB 9100
5678 IF PA(3)=–999 THEN PA(3)=0
5680 PA(3)=PA(3)+X
5685 PRINT INT(PA(3)/(CN*PN(3)));"SEMANAS"
5690 GOTO 5530
5700 REM EVENTO 7 – RECOGER AGUA
5705 PRINT
5710 S$="    DURANTE LA SEMANA*":GOSUB 9100
5715 PRINT:GOSUB 9200
5720 S$="UN TEMPORAL DE LLUVIA LLENO TUS*":GOSUB 9100
5725 S$="BARRILES DE AGUA*":GOSUB 9100
5730 PRINT:GOSUB 9200
5735 X=INT(RND(1)*10)+11
5736 REM ENTRE 10 y 20 BARRILES
5740 S$="AHORA TU PROVISION DE AGUA ES*":GOSUB 9100
5745 S$="SUFICIENTE PARA APROXIMADAMENTE*":GOSUB 9100
5748 IF PA(4)=–999 THEN PA(4)=0
5750 PA(4)=PA(4)+X
5755 PRINT INT(PA(4)/(CN*PN(4)));"SEMANAS"
5760 GOTO 5530
5800 REM EVENTO 8 – VIENTOS FAVORABLES
5805 PRINT
5810 S$="FUERTES VIENTOS CONTINUOS DURANTE TODA LA
     SEMANA*":GOSUB 9100
5815 PRINT:GOSUB 9200
5820 S$="HAS NAVEGADO A BUENA VELOCIDAD*":GOSUB 9100
5825 S$="Y LA DURACION DEL VIAJE SE*":GOSUB 9100
5830 S$="REDUCE EN MEDIA SEMANA*":GOSUB 9100
5835 EW=EW–.5
5839 GOTO 5530
5850 REM EVENTO 9 – BUEN TIEMPO
5855 PRINT
5860 S$="BUEN TIEMPO DURANTE TODA LA SEMANA*":GOSUB
     9100
5865 PRINT:GOSUB 9200
5870 S$="LA TRIPULACION SE SIENTE MAS FELIZ*":GOSUB 9100
5875 GOSUB 9200
5880 S$="    ¡Y ESTA MAS SALUDABLE!*":GOSUB 9100
5882 FOR T=1 TO 16
5885 IF TS(T,2)=0 OR TS(T,2)=–999 THEN 5888
5886 TS(T,2)=TS(T,2)+INT(RND(1)*11)+5
5888 NEXT
5889 GOTO 5530
5900 REM EVENTO 10 – PERDIDA DE MEDICINAS
5905 IF OA(1)=0 OR OA(1)=–999 THEN RETURN
5910 PRINT
5915 S$="DESCUBRES QUE LA MITAD DE TUS*":GOSUB 9100
5920 S$="FRASCOS DE MEDICINA SE HAN ROTO*":GOSUB 9100
5925 OA(1)=INT(OA(1)/2)
5930 PRINT:GOSUB 9200
5935 S$="AHORA TE QUEDAN*":GOSUB 9100
5940 PRINT OA(1);"FRASCOS SOLAMENTE"
5945 GOTO 5530
5950 REM EVENTO 11 – ARMAS OXIDADAS
5955 IF OA(2)=0 OR OA(2)=–999 THEN RETURN
5960 PRINT
5965 S$="DESCUBRES QUE LA MITAD DE TUS*":GOSUB 9100
5970 S$="ARMAS SE HAN OXIDADO*":GOSUB 9100
5972 S$="Y YA NO SIRVEN*":GOSUB 9100
5975 OA(2)=INT(OA(2)/2)
5980 PRINT:GOSUB 9200
5985 S$="AHORA TE QUEDAN*":GOSUB 9100
5990 PRINT OA(2);"ARMAS SOLAMENTE"
5995 GOTO 5530
6000 REM EVENTO 12 – PERDIDA DE TELA
6005 IF OA(4)=0 OROA(4)=–999 THEN RETURN
6010 PRINT
6015 S$="DESCUBRES QUE LOS RATONES*":GOSUB 9100
6020 S$="SE HAN COMIDO LA MITAD DE TUS*":GOSUB 9100
6022 S$="BALAS DE TELA Y AHORA*":GOSUB 9100
6024 S$="ESTAS YA NO SIRVEN*":GOSUB 9100
6025 OA(4)=INT(OA(4)/2)
6030 PRINT:GOSUB 9200
6035 S$="AHORA TE QUEDAN*":GOSUB 9100
6040 PRINT OA(4);"BALAS SOLAMENTE"
6045 GOTO 5530
6050 REM EVENTO 13 – ALBATROS
6055 PRINT:A$="S"
6060 S$="UN ALBATROS SOBREVUELA EL BARCO*":GOSUB 9100
6062 GOSUB 9200
6065 S$="ESTE ES UN BUEN PRESAGIO*":GOSUB 9100
6068 S$="Y LA TRIPULACION ESTA CONTENTA*":GOSUB 9100
6070 PRINT:GOSUB 9200
6075 IF PA(3)<(CN*PN(3)*(JL–WK+1)) THEN 6090
6080 REM NO HAY ESCASEZ DE CARNE
6085 GOTO 6122
6090 S$="SE TE ESTA TERMINANDO LA CARNE*":GOSUB 9100
6095 S$="Y EL AVE PESA 10 KILOS!*":GOSUB 9100
6100 PRINT:GOSUB 9200
6105 S$="TE GUSTARIA COGERLA?*":GOSUB 9100
6110 INPUT I$
6112 PRINT:GOSUB 9200
6115 IF LEFT$(I$,1)="S" THEN 6133
6120 S$="PROBABLEMENTE DA LO MISMO!*":GOSUB 9100
6122 PRINT:GOSUB 9200
6125 S$="EL ALBATROS SE ALEJA VOLANDO*":GOSUB 9100
6130 GOTO 5530
6133 IF OA(2)=0 OR OA(2)=–999 THEN 6180
6135 S$="HACES UN DISPARO...........*":GOSUB 9100
6138 GOSUB 9200:GOSUB 9200
6140 IF RND(1)<.5 THEN 6150
6145 S$="...............PERO NO ACIERTAS!*":GOSUB 9100
6148 GOTO 6122
6150 S$="Y EL AVE CAE SOBRE LA CUBIERTA!*":GOSUB 9100
6155 PRINT:GOSUB 9200
6160 IF PA(3)=–999 THEN PA(3)=0
6162 PA(3)=PA(3)+10:B$="S"
6165 S$="AHORA TIENES 10 KILOS MAS*":GOSUB 9100
6167 S$="DE CARNE......*":GOSUB 9100
6170 S$="PERO PUEDE SER QUE LA BUENA FORTUNA*":GOSUB
     9100
6172 S$="NO TE ACOMPAÑE A PARTIR DE AHORA!!*":GOSUB
     9100
6174 GOTO 5530
6180 S$="NO PUEDES...NO TIENES ARMAS*":GOSUB 9100
6190 GOTO 6122
```

Complementos al BASIC

Spectrum:

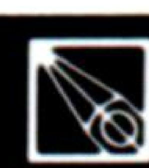
Introduzca las siguientes modificaciones:

```
875  IF HR(3)=.5 AND RND(1)<.5
     THEN CLS:GO SUB 6050
5527 IF X=6 THEN GO TO 5600
5528 IF X=7 THEN GO TO 5700
5529 IF X=8 THEN GO TO 5800
5530 IF X=9 THEN GO TO 5850
5531 IF X=10 THEN GO TO 5900
5532 IF X=11 THEN GO TO 5950
5533 IF X=12 THEN GO TO 6000
5534 IF X=13 THEN GO TO 6050
5535 PRINT:S$=K$:GO SUB 9100
5536 LET I$=INKEY$:IF INKEY$=""THEN
     GO TO 5536
6115 IF I$(1 TO 1)="S" THEN GO TO 6133
```

BBC Micro:

Introduzca las siguientes modificaciones:

```
875  IF HR(3)=.5 AND RND(1)<.5
     THEN CLS:GOSUB 6050
```



A prudente distancia

Nos corresponde diseñar el primer fragmento de software para controlar los movimientos del brazo-robot

Los cuatro motores que mueven el brazo están conectados a las cuatro líneas de datos del ordenador con las que está conectado el brazo en interface, que son también las que controlan los motores. En el caso del BBC Micro, estas cuatro líneas de datos parten de la puerta para el usuario utilizando las patillas de D0 a D3. Debido a que los servomotores se basan en un flujo continuo de impulsos que les indican qué ángulo seguir, no se los puede controlar exclusivamente desde BASIC, sino que deben ser activados mediante un programa cuña en código máquina, que envíe un nuevo grupo de señales a los motores una vez cada dieciseisavo de segundo utilizando una interrupción. (Anteriormente ya hemos explicado detalladamente los principios de este método de control.) Se puede emplear el programa de control para servomotores múltiples ofrecido con anterioridad para comprobar la operación del brazo directamente desde el teclado y, si bien no es muy sofisticado, probará cada motor de forma independiente.

Una vez que el código cuña esté ejecutándose como tarea de fondo, será posible salir del programa de exploración de teclas del BASIC o colocar números directamente en las posiciones que controlan los ángulos de los motores. La dirección de la primera posición de memoria motor-ángulo es ángulo; motor 0 se establece utilizando el valor de esta posición. La posición ángulo+1 establece la posición del motor 1, y así sucesivamente. Recuerde que el motor 0 controla el movimiento de la cintura, el motor 1 el movimiento del hombro, y el motor 3 abre y cierra el mecanismo de prensión. Si cualquier ángulo se incrementa o disminuye en una unidad, los motores lo pueden seguir fácilmente. Si se introducen cambios grandes, como de 0 a 128, entonces los motores pueden tardar unos instantes en reaccionar. Asimismo, observe cómo los efectos combinados de la inercia y "esponjosidad" del brazo hacen que éste se detenga de forma brusca, que recuerda la moda de la danza robótica.

Tras probar el brazo utilizando el sencillo programa de control, se hace evidente la necesidad de un programa más sofisticado. No obstante, decidir exactamente cómo programar el brazo probablemente sea la parte más difícil de toda la operación. Muchos robots industriales se programan moviendo la pinza manualmente a través de la secuencia que debe aprender el robot; el brazo graba de manera automática las posiciones clave de la secuencia. Lamentablemente, este método exige sofisticados elementos de hardware, como sensores de realimentación angular, de los que carece nuestro robot.

Pero incluso con equipos sofisticados, tales como los empleados para aplicaciones industriales, el

Ingeniería estructural

Nivel 0
Programador de secuencias para el brazo
Nivel 1
Preparar y ensamblar código máquina
Menú principal
Nivel 2
Iniciar temporizador
Opción 1
Opción 2
Opción 3
Opción 4
Nivel 3
Contador =0
REPEAT...UNTIL G$="Q" OR cont. >maxcont.
Replay
Save
REPEAT... UNTIL G$="Q" OR cont.> maxcont.
Replay
Save
Cargar archivo
Replay
Detener temporizador
Informe y programa
Informe y programa

El diagrama ilustra la estructura "activada por menú" del programador de secuencias para el brazo con el BBC Micro. Las opciones 1 y 2 difieren sólo en que la opción 1 restablece el contador, borrando el contenido anterior de la matriz de posiciones de teclas, R%(,). La opción 3 vuelve a cargar un archivo de posiciones de teclas y las reproduce. La opción 4 detiene el temporizador y hace terminar el programa

Kevin Jones

brazo podría trabajar en entornos peligrosos en los que un operador no podría físicamente programar el brazo. En consecuencia, un método alternativo consiste en controlar el brazo a distancia desde el teclado, haciéndolo pasar por una serie de movimientos y guardando las posiciones clave de la secuencia mediante la pulsación de otra tecla, y registrando los cuatro ángulos de motor que definen cada posición en una matriz. La secuencia se puede reproducir simplemente recorriendo la matriz de posiciones de teclas con un factor de retardo adecuado. Las secuencias también se pueden guardar o cargar desde cinta o disco empleando un archivo secuencial para almacenar el contenido de la matriz. Éste es el procedimiento que utilizaremos.

La primera etapa del diseño de nuestro software es, por lo tanto, asignar teclas para controlar las diversas funciones del brazo, de modo que pueda "enseñársele" una secuencia de movimientos moviéndolo a distancia desde el teclado. Se pueden reasignar las teclas del cursor del BBC Micro para que controlen el movimiento izquierda/derecha del cuerpo principal y el movimiento arriba/abajo del brazo inferior. Las teclas A y Z se utilizarán para mover el brazo superior hacia arriba y hacia abajo, y las teclas X y C para abrir y cerrar las garras de la pinza. Las teclas I y D permiten aumentar o reducir la velocidad del brazo, de modo que las maniobras delicadas se puedan llevar a cabo lentamente, mientras los movimientos de barrido amplios se realizan con rapidez. La tecla S permite almacenar la posición actual del brazo en la matriz de posiciones de teclas, e Y retorna el brazo a la última posición almacenada. La secuencia se puede realizar para introducir cambios, ya sea hacia adelante empleando la tecla N, o bien hacia atrás, pulsando la tecla B; E controla movimientos a ciertos puntos de la secuencia. Utilizando estas tres últimas teclas, se puede editar cualquier secuencia de posiciones programada.

Ian McKinnell

Recogiendo las piezas
Comenzamos el proyecto del brazo-robot considerando una tarea determinada: recoger y trasladar un objeto del tamaño y peso de una carrete de hilo de coser. El programador de secuencias nos permite programarlo para que realice esta tarea, guardando los movimientos, si fuera necesario, en un archivo que se puede volver a cargar y reproducir

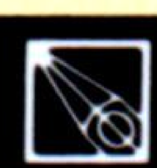
Los procedimientos informe y programa del listado le proporcionan una lista de las funciones de las teclas, además de explorar repetidamente el teclado en busca de una entrada.

Los otros procedimientos, replay y save, permiten reproducir o guardar en cinta o disco la secuencia retenida en la matriz de posiciones de teclas, R%(,), mientras que cargararchivo reasigna los valores de R%(,) utilizando valores previamente guardados en un archivo secuencial. Por tanto, tomado como un todo, el programa nos ofrece un método cabal para programar, editar y reproducir secuencias de movimientos del brazo.

Un movimiento más uniforme

La rutina en código máquina para manipulación de interrupciones controla los ángulos de los servomotores analizando cuatro posiciones, que comienzan en la dirección ángulo. Cada una de estas posiciones contiene un valor que corresponde al ángulo que debe asumir cada motor. Una de las razones por las cuales el programa de prueba original produce movimientos discontinuos es que el programa en BASIC coloca (POKE) valores directamente en estas posiciones. Los grandes cambios en un ángulo producen un movimiento violento del motor, dado que éste intenta alcanzar la nueva posición con la mayor rapidez posible. Para conseguir que el brazo se desplace de forma más uniforme, deberíamos tratar de ajustar los valores de las cuatro posiciones ángulo sólo en pequeñas cantidades. Por este motivo se utiliza un segundo grupo de posiciones, que comienzan en nuevapos%, para aceptar los cambios de ángulo desde el programa en BASIC. Se puede añadir luego al programa una corta rutina en código máquina para aumentar o disminuir los valores de las cuatro posiciones ángulo (en pasos de una unidad) hasta que concuerden con los valores de la correspondiente posición nuevapos%. El procedimiento moverservo simplemente llama al programa en código máquina que hace esto.

La introducción de esta nueva rutina ofrece una ventaja adicional. Mediante la inserción de un bucle de demora dentro de este trozo de código máquina, podemos retardar o acelerar la velocidad a la cual se mueve el motor desde su antigua posición a su nueva posición, simplemente alterando el valor que da por terminado el bucle de demora. La posición &81 almacena este factor de retardo, que se puede alterar desde el BASIC al comienzo del procedimiento replay, permitiendo implementar a diferentes velocidades las secuencias de movimientos programados previamente. El código máquina para esta nueva rutina comienza en la línea 2000 del listado que ofrecemos.



Prog. secuencias BBC Micro

```
  10 REM ******************************
  11 REM ******************************
  12 REM **                          **
  13 REM **       PROGRAMADOR        **
  14 REM **   SECUENCIAS BRAZO BBC   **
  15 REM **                          **
  16 REM ******************************
 .17 REM ******************************
  20 :
  25 MODE 3
  30 PROCpreparacion
  40 CLS:PRINT TAB(5,6)"Te gustaria?:"
  50 PRINT TAB(25,10)"1 .................................. Programar nueva secuencia para brazo"
  60 PRINT TAB(25,12)"2 .................................. Añadir movimientos al programa"
  70 PRINT TAB(25,14)"3 .................................. Reproducir un archivo"
  80 PRINT TAB(25,16)"4 .................................. Salir del programa"
  90 G$=GET$:IF G$="1" THEN contador=0
 100 IF G$="1" OR G$="2" THEN PROCinforme:PROCprograma:PROCreplay:PROCsave
 110 IG G$="3" THEN PROCcargararchivo:PROCreplay
 120 IF G$="4" THEN CLS:PROCfintemporizador:END
 130 GOTO 40
 140
 150
 160 DEF PROCinforme:CLS
 170 PRINT TAB(5,5)"Por favor utiliza  :"
 180 PRINT TAB(20,7)"Teclas cursor...para I/D y 1.er brazo Ar/Ab"
 190 PRINT TAB(20)"A & Z ................................ para 2do brazo Ar/Ab"
 200 PRINT TAB(20)"X & C................................. para movimiento Abrir/Cerrar pinza"
 210 PRINT TAB(20)"'S' .................................. para guardar posiciones servos"
 220 PRINT TAB(20)"'Q' .................................. para regresar al Menu"
 230 PRINT TAB(20)"'R' .................................. mover a posicion guardada"
 240 PRINT TAB(20)"N & B................................. siguiente y anterior contador.
                                                          E .. nuevo contador"
 250 PRINT TAB(20)"I & D ................................ aumentar y reducir velocidad"
 260 PRINT TAB(2,2)"contador="contador"     ";:FOR I%=0 TO 3:PRINT R%
     (contador,I%);"     ";:NEXT:PRINT
 270 ENDPROC
 280
 290
 300 DEF PROCprograma:?&81=1
 310 REM reasignar teclas movimiento cursor
 320 *FX4,1
 330 DX=2
 340 REPEAT :REM explorar teclas
 350   IF INKEY(-38)THEN DX=DX+1    :REM "I"
 360   IF INKEY(-51)THEN DX=DX-1    :IF DX<1 THEN DX=1:REM "D"
 380   IF INKEY(-26)THEN nuevapos%?0=(nuevapos%?0)+DX:REM "FLECHA IZQUIERDA" servo 0 derecha
 390   IF INKEY(-122) THEN nuevapos%?0=(nuevapos%?0)-DX: REM "FLECHA DERECHA" servo 0 izquierda
 400   IF INKEY(-58)THEN nuevapos%?1=(nuevapos%?1)+DX: REM "FLECHA ARRIBA" servo 1 arriba
 410   IF INKEY(-42)THEN nuevapos%?1=(nuevapos%?1)-DX: REM "FLECHA ABAJO" servo 1 abajo
 420   IF INKEY(-66)THEN nuevapos%?2=(nuevapos%?2)-DX: REM "A" servo 2 arriba
 430   IF INKEY(-98) THEN nuevapos%?2=(nuevapos%?2)+DX: REM "Z" servo 2 abajo
 440   IF INKEY(-67) THEN nuevapos%?3=(nuevapos%?3)+DX*3: REM "X" servo 3 abrir pinza
 450   IF INKEY(-83) THEN nuevapos%?3=(nuevapos%?3)-DX*3: REM "C" servo 3 cerrar pinza
 460   IF INKEY(-52) THEN FOR I%=0 TO 3:nuevapos%?I%=R%(contador,I%): NEXT:REM "R" reproducir posicion
 470   IF INKEY(-86)THEN contador=contador+1:IF contador>maxcontador THEN contador=0
 480   IF INKEY (-101)THEN contador=contador-1:IF contador<0 THEN contador=maxcontador
 490   IF INKEY(-35)THEN PRINTTAB(10,20)"DAR VALOR PARA contador     ";: INPUT contador :REM "E" entrar nuevo
       contador
 500   IF INKEY(-82)THEN FOR I%=0 TO
       Ningunservo%:R%(contador,I%)=nuevapos%?I%:NEXT:contador=contador+1:REM "S" guardar posicion en
       contador actual
 510   IF contador>ultimomaxcont THEN ultimomaxcont=contador
 520   IF antcont<>contador THEN PRINTTAB(2,2)"contador="contador"     ";:FOR I%=0 TO 3:PRINT
       R%(contador,I%);"     ";:NEXT:PRINT
 530   antcont=contador
 540 FOR I%=0 TO 3:IF nuevapos%?I%>200 THEN nuevapos%?I%=200
 550 IF nuevapos%?I%<10 THEN nuevapos%?I%=10
 560 NEXT
 570 PROCmoverservo
 580 G$=INKEY$(1):UNTIL G$="Q"OR contador>maxcontador
 590 *FX4,0
 600 ENDPROC
 610
 620
 630 DEF PROCreplay
 640 REPEAT
 650 PRINTTAB(11,23)"Quieres reproducir movimientos? (S/N) R para repetir"
 660 G$=GET$
 670 UNTIL G$="S" OR G$="N" OR G$="R"
 680 IF G$="N" THEN ENDPROC
 690 IF G$<>"R" THEN INPUTTAB(11,22)"Que periodo de demora entre respuesta? (1 a 255) despues <return>";dly%:IF
     dly%>255 OR dly%<1 GOTO 690
 700 ?&81=dly%
 710 FOR NumPos=0 TO ultimomaxcont:PRINTTAB(19,2)"Num. de posicion secuencial   =   ";NumPos;"     ":FOR
     servo%=0 TO Ningunservo%:nuevapos%?servo%=R%(NumPos,servo%):NEXT:PROCmoverservo:NEXT
 720 GOTO 640
 730
 740
 750 DEF PROCmoverservo
 760 CALLmoverservo :REM version codigo maquina
 820 ENDPROC
 830
 831
 840 DEF PROCsave:CLS
 850 PRINT TAB(15,10)"Te gustaria guardar la secuencia en un archivo? S/N  ":IF GET$<>"S" THEN ENDPROC
 860 INPUT TAB(15,12)"Por favor nombra el Archivo a guardar  :  ";archivo$ ' ' :archivo$="D."+archivo$
 870 PROCfintemporizador
 880 X=OPENOUT archivo$:PRINT#X,ultimomaxcont:FOR A=0 TO ultimomaxcont:FOR B=0 TO
     Ningunservo%:PRINT#X,R%(A,B):NEXT:NEXT
 890 CLOSE#X
 900 PROCiniciartemporizador
 910 ENDPROC
 920
 930
 940 DEF PROCcargararchivo:CLS
 950 INPUT TAB(15,10)"Por favor entra el nombre del archivo a cargar:";archivo$:archivo$="D."+archivo$
 960 PROCfintemporizador
 970 X=OPENIN archivo$:INPUT #X,contador:ultimomaxcont=contador:FOR A=0 TO contador:FOR B=0 TO
     Ningunservo%:INPUT#X, R%(A,b):NEXT:NEXT
 980 antcont=0
 990 CLOSE#X
1000 PROCiniciartemporizador
1010 ENDPROC
1020
1030
1040 DEF PROCpreparacion
1050 contador=0:Ningunservo%=3:REM NUM. DE MOTORES-1
1060 ultimomaxcont=0  :antcont=0  :dly%=10
1070 maxcontador=100
1080 DIM R%(maxcontador,Ningunservo%),nuevapos% 8,temporizador% 12, leer% 12
1090 PROCassembletime
1100 ENDPROC
1110
1120
1130 DEF PROCassembletime
1140 REM ***************************************************
1150 REM                 Establecer temporizador etc.
1160 REM ***************************************************
1170 osbyte=&FFF4
1180 A%=&97  :X%=&62  :Y%=&FF
1190 CALL osbyte:REM establecer puerta B para salida
1200 DIM p%(8)
1210 DIM temporizador% 12,leer% 12
1220 xtemporizador=temporizador% MOD 256
1230 ytemporizador=temporizador% DIV 256
1240 xleer=leer% MOD 256
1250 yleer=leer% DIV 256
1260 PROCinicial
1270 FOR I%=angulo TO angulo+8:angulo?I%=128:NEXT
1280 t=.02  :REM seg entre impulsos
1290 tiempo%=&FFFFFFFF-(t*100)+1
1300 temporizador%?4=&FF    :REM cargar byte mas alto
1310 !temporizador%=tiempo%:REM establecer temporizador, habilitar eventos
1320 PROCiniciartemporizador
1330 ENDPROC
1340
1350 DEF PROCiniciartemporizador
1360 *FX14,5
1370 A%=4    :X%=xtemporizador    :Y%=ytemporizador    :CALL &FFF1
1380 ENDPROC
1390
1400 DEF PROCfintemporizador
1410 *FX13,5
1420 ENDPROC
1430
1440
1450 DEF PROCinicial
1460   REM ***************************************************
1470   REM                Ensamblar el codigo maquina
1480   REM ***************************************************
1490 DIM espacio% 500
1500 FOR C=0 TO 3 STEP 3
1510 paginacero=&70    :REM libre para usuarios
1520 puertab=&FE60    :osword=&FFF1
1530 P%=espacio%
1540 angulo=P%  :P%=P%+8  :REM potencialmente 8 motores
1550 tabla=P%  :P%=P%+256  :REM 256 longitudes de impulso posibles
1560 FOR I%=tabla TO tabla+&100:?I%=&FF:NEXT
1570 lowtabla%=tabla MOD 256
1580 hightabla%=tabla DIV 256
1590 ?paginacero=lowtabla%  :paginacero?1=hightabla%
1600 [OPT C
1610 .manipuladoreventos
1620 PHP:PHA:TYA:PHA:TXA:PHA
1630 LDA #&04
1640 LDX #xtemporizador
1650 LDX #ytemporizador
1660 JSR osword
1670 LDY #&0
1680   /Empezar impulso, para algunos motores seria posible empezar
1690   /antes de llenar la tabla y reducir asi el bucle de espera de abajo
1700 LDA #&FF  :STA puertab
1710   /llenar tabla con excepciones
1720 LDX #&7  :LDA #FF  :CLC /preparar patron bits
1730 .excepciones
1740       ROR A  :PHA /patron bits
1750       LDY angulo,X /tomar despl. corresp. a angulo motor X
1760       AND (paginacero),Y /conservar patron bits existente
1770       STA (paginacero),Y /pero modificado para motor X
1780       PLA :DEX
1790       BPL excepciones
1800 /ahora la tabla esta cargada, se llenara en poco tiempo
1810 LDY #&60
1820 .esperar      DEY
1830               BNE esperar
1840 LDA #&FF /todos los impulsos encendidos
1850 LDY #&0
1860 .bucle AND (paginacero), Y/pero desenmascarar con cada
1870               STA puerta b /de la tabla de uno en uno
1880               INY
1890               BNE bucle
1900 LDX #&7  :LDA #&FF
1910 .borrar
1920       LDY angulo,X /borrar otra vez todas las excepciones
1930       STA (paginacero),Y
1940 DEX
1950 BPL borrar
1960 /todos los impulsos han de haber acabado ya
1970 PLA:TAX:PLA:TAY:PLA:PLP
1980 RTS
1990
2000
2010 .moverservo /......movimiento uniforme cod. maq. ......
2020 .Bucle2
2030 LDY #0
2040 LDY #0
2050 .Bucle1
2060    LDA nuevapos%,X          /   nueva posicion para servo x
2070    CMP angulo,X             /   es la antigua posición la misma?
2080    BEQ movido               /   if=no hacer nada
2090       BSC sumar                 if>sumar
2100          DEC angulo,X  /   else sacar uno
2110          JMP incrementarx
2120         .sumar
2130            INC angulo,X
2140          JMP incrementarx
2140       .movido
2150          INY
2170     .incrementarx
2180     INX  :CPX #4
2190     BNE Bucle1              /   he comprobado los 4 servos?
2200 JSR esperarmotores
2210 CPY #4
2220 BNE Bucle2                  /   continuar si no han terminado todos
2230 RTS                         /  al llegar aqui todos los servos en pos correcta
2240
2250
2260 .esperarmotores
2270 TXA:PHA:TYA:PHA
2280 LDY #&FF  :LDX &81
2290 .esperando
2300    DEY    :NOP  :NOP
2310    BNE esperando            /   bucle 8 ciclos reloj, o sea 1024 ciclos
2320 LDY #&FF
2330 DEX
2340 BNE esperando
2350 PLA:TAY:PLA:TAX
2360 RTS
2370
2380 ]
2390 NEXT
2400 !&220=manipuladoreventos OR (!&220 AND &FFFF0000)
2410 ENDPROC
```

Lenguaje estructural

En PASCAL es posible definir procedimientos y funciones para complementar aquellos que ya están incorporados en el lenguaje

Antes de dar por finalizado nuestro informe sobre las estructuras de datos del PASCAL, echaremos una mirada a algunos mecanismos para estructurar programas. Ello se realiza definiendo nuestros propios procedimientos y funciones para complementar aquellos que ya se hallan incorporados en el lenguaje PASCAL, tales como el procedimiento write y la función sqrt. Pero, antes de definir algunos por nuestra propia cuenta, veamos por qué write es un procedimiento y no una función, y por qué sqrt es una función y no un procedimiento. Cuando decimos, por ejemplo:

```
write ('Hola!')
```

esperamos que la serie de caracteres proporcionados como el parámetro simple aparezca en el archivo de salida estándar. En otras palabras, write identifica un subprograma que sabe cómo enviar datos en forma de una corriente de caracteres al dispositivo de salida (VDU). Pero ¿qué haría la siguiente "sentencia"?

```
sqrt(256)
```

La respuesta, por supuesto, es que generaría un mensaje de error de tiempo de compilación: no es una sentencia legal. Por otra parte:

```
write (sqrt(256))
```

no sólo sería legal, sino que produciría algo así como 1.60000E+01 en la pantalla. Pregúntese a sí mismo la razón, y por qué parece ser tan inevitable y obvio; si puede darse cuenta de los motivos, no albergará ningún tipo de duda sobre la diferencia fundamental entre un procedimiento y una función.

El identificador write invoca a un proceso que envía datos a una corriente de salida y es un procedimiento al cual se puede llamar por su nombre. Otro procedimiento del PASCAL es page (F), que se emplea para comenzar una nueva página en el archivo de textos F. El identificador sqrt, sin embargo, siempre debe proporcionarse con un "argumento" numérico simple y simplemente devuelve un resultado: el número real que es la raíz cuadrada de su argumento. De modo que la primera distinción entre procedimientos y funciones es que no existe una sentencia de función como tal. Así como escribir 16 de forma aislada no significa nada (no es más que un valor), las expresiones tales como sqrt(X/3) u odd(N) no poseen ningún significado propio. Los identificadores de funciones se comportan como variables que no han sido inicializadas, pero cuyo valor se calcula cada vez que se utiliza su nombre en una sentencia de programa. El resultado de valor simple que se devuelve es determinado a partir de una inspección del valor (o valores) actual del o los argumentos de la función; es decir, el resultado es una función del o los argumentos.

Los procedimientos, por el contrario, no devuelven un valor y, por tanto, no se pueden utilizar en expresiones. Esto no es sorprendente. La sentencia N:=WriteLn demuestra a las claras su ilegalidad, tal como lo sería LET N=PRINT en BASIC.

La tarea de escribir programas muy extensos en PASCAL se simplifica gracias a la capacidad para otorgar nombres a nuestros propios procedimientos y funciones y controlar su accesibilidad y la relación entre ellos. Tal como sucede con los nombres que se les puede dar a las variables, la relativa carencia de restricciones respecto a los identificadores significa que para nuestros subprogramas podemos elegir nombres útiles que tengan algún significado. Consideremos el siguiente problema:

```
PROGRAM Incompleto(input,output,archivodatos);
                    {declaraciones..}
BEGIN
   Abrir(archivodatos);
   WHILE NOT EoF (archivodatos)DO
      BEGIN
         read (archivodatos,item);
         Procesar(item)
      END
END.
```

Aun cuando todavía no hayamos visto el tratamiento de archivos, usted no tendrá ninguna dificultad en comprender qué es lo que hace el programa. En la parte de declaraciones, declararíamos un procedimiento para localizar un archivo de datos en el almacenamiento de apoyo y abrirlo para leerlo (Abrir) y otro para tratar cada elemento de datos de la forma adecuada (Procesar). El programa utiliza, asimismo, dos de los identificadores predefinidos del PASCAL. El procedimiento read (que hasta ahora hemos utilizado para leer entradas de texto) se puede emplear, como podemos ver, para leer datos, estructurados o no, de un archivo del tipo apropiado. La función booleana EoF (*End of File*: final del archivo) devuelve el valor "verdadero" tras haberse leído el último registro del archivo de su argumento. Nuevamente, obviando el identificador del archivo, pasa por defecto a la entrada del archivo estándar. Sólo para los archivos de texto, la función EoLn (*End of Line*: final de línea) devuelve "verdadero" cuando se ha leído el último carácter de una línea. Veamos cómo se aplica todo esto cuando definimos nuestros propios subprogramas.

En esencia, existe poca diferencia entre programas, procedimientos y funciones; todos son módulos que poseen sus propias descripciones de datos locales y cuerpos de sentencias. La única variación de sintaxis se produce en el encabezamiento. Un encabezamiento de programa, como ya hemos visto, se compone de la palabra reservada PROGRAM, un identificador del usuario que da nombre



al programa y una lista de parámetros que identifican los archivos con los cuales se comunicará el programa. El encabezamiento de un procedimiento sustituye la palabra reservada por PROCEDURE y amplía la "lista formal de parámetros" para incluir los identificadores del tipo de cada parámetro de datos. De modo que, por ejemplo:

```
PROCEDURE Lineas (NumLineas : integer);
  VAR
    N : integer;
BEGIN
  FOR N:=1 TO NumLineas DO
    WriteLn
END;
```

El END del procedimiento va seguido de un punto y coma, y no de un punto, como el END del final de un programa. A NumLineas, el identificador formal del parámetro, se le pasa el valor concreto en la sentencia de llamada:

```
Lineas (5)
```

Este procedimiento indudablemente trivial es, sin embargo, de mucha utilidad si con frecuencia deseamos dejar varias líneas en blanco (cinco, en este caso) para que la salida sea más clara, y ahorrarnos (a nosotros y al compilador) el tener secuencias más bien aburridas de sentencias WriteLn separadas por puntos y coma. Este procedimiento de propósito general le permitirá dejar en blanco tantas líneas como le sea necesario, de modo que Lineas(10) producirá 10 líneas en blanco, Lineas(20) producirá 20, y así sucesivamente.

Este ejemplo es, asimismo, una buena ilustración de la seguridad del PASCAL. La variable de control del bucle FOR siempre se *debe* declarar como una variable local. No era posible, por ejemplo:

```
For NumLineas := 1 TO NumLineas DO...
```

Aunque esto sería aceptable en un programa principal, no se puede garantizar la seguridad del bucle si un controlador de bucle no es local o "relativamente global". Quizá usted alguna vez se pasó horas depurando un programa en BASIC que contenía una sentencia como ésta:

```
300 FOR N=1 TO T
400 GOSUB 2000
500 NEXT N
```

¡para descubrir, finalmente, que una subrutina remota, tal vez llamada indirecta y condicionadamente, utilizaba N para algún otro fin! El PASCAL prohíbe expresamente un comportamiento tan incontrolado como éste, y ayuda a ahorrar el tiempo desperdiciado en el desarrollo. Todo identificador declarado dentro de un bloque posee una "región" definida que se extiende a través de ese bloque. Sin embargo, su esfera de acción o su "ámbito" (la parte del bloque desde la cual es accesible) sólo se extiende desde el punto de su declaración hasta el final de ese bloque, y esto se puede limitar aún más mediante la redefinición. En el ejemplo anterior, la N a la que se alude en el procedimiento Lineas es, naturalmente, el entero declarado localmente, y esta declaración anula temporalmente cualquier otra relativamente global durante cada activación del procedimiento.

Refiriéndonos al programa principal (Ambito) del ejemplo, la región de las variables globales N y X es la totalidad del programa. La esfera de acción de X comprende desde su declaración hasta el final del programa, excluyendo cualquier otro sitio dentro del procedimiento A. Ello se debe a que se define otra X (SET OF char) como un identificador de tipo y su región es el bloque de A. De modo similar, dentro de B, N alude al parámetro formal de B, y no a la variable global.

Aunque la región tanto de A como de B es la totalidad del programa, la esfera de acción de B sólo empieza en su punto de definición y, por tanto, si bien usted podría utilizar A en el procedimiento B, B es invisible para A. Ello refuerza la lógica cáustica del PASCAL: ¡no se puede ejecutar ningún programa hasta que no haya sido escrito! Esto explica, asimismo, por qué las definiciones y las declaraciones *deben* respetar el orden CONST, TYPE, VAR, seguidas de las definiciones de procedimiento y función ordenadas de acuerdo a las exigencias estructurales del programa. Muchos compiladores de PASCAL son "de un solo paso" (sólo leen el código fuente una vez), y esto no sería posible si no se respetara este orden lógico.

El esfuerzo adicional que supone el tener que declarar los datos locales a cada procedimiento se recompensa con creces. Se obtiene modularidad gracias al paso de todos los valores de datos como parámetros a procedimientos, y si bien quizá usted vea programas en PASCAL que utilizan procedimientos sin ninguna lista de parámetros, accediendo globalmente a todos los datos, es recomendable evitar estrictamente esta práctica. En realidad el hecho de que permita esta clase de abusos se puede considerar como uno de los puntos débiles del PASCAL. Observe que, así como cualquier referencia a X dentro de A significa el tipo X, la utilización de N dentro de N alude a su parámetro formal local y el entero global N queda temporalmente inaccesible. Aparte de la seguridad inherente, esto permite que un equipo de programadores trabajen juntos en un proyecto grande sin tener que preocuparse por conflictos de identificadores.

Muchísimo alcance

```
PROGRAM Alcance;
VAR
      N   : integer;
      X   : real;
PROCEDURE A(Y:real);
  TYPE
    X=SET OF char;
  {etc.}
PROCEDURE B (N:integer);
  {etc.}
BEGIN {Programa principal
  -Alcance}
  "
  ...
  A(succ(N)/3);
  B(N);
  A(X)
    {etc.}
END.
```

En el programa Alcance, que esbozamos aquí, las regiones y las esferas de acción de las diferentes variables y procedimientos se reflejan en la siguiente tabla:

Proc/Var	Región	Ambito (alcance)
A	principal	A, B, princ.
B	principal	B, principal
N (global)	principal	A, principal
X (real)	principal	B, principal
X (en proc A)	A	A
N (en proc B)	B	B

La llamada a B pasa el valor de la global N como un parámetro real, que resulta tener el mismo nombre que en B pero que es una variable completamente separada. Ahora conviene puntualizar:

- Al entrar en un bloque se realiza una copia local del valor pasado.
- Cualquier cambio de este "parámetro de valor" dentro de su bloque no incide en el parámetro real que se pasa desde el punto de llamada.
- El valor pasado puede ser cualquier expresión del tipo correcto.

Naturalmente, cualquier sentencia de procedimiento debe listar los parámetros reales de la llamada, y éstos deben corresponderse con la lista de parámetros formales del encabezamiento en número, posición y tipo. El mecanismo de paso de parámetro por valor se puede considerar como la inicialización del identificador del parámetro formal al valor, cualquiera que fuera, especificado en la sentencia de llamada. De modo que:

```
Lineas (succ(N+vacio)DIV 2)
```

implica la siguiente sentencia de asignación al entrar en Lineas:

```
NumLineas := succ(N+vacio)DIV 2
```

Teniendo presente esta forma segura de pasar valores, quizá le interese imaginar cómo podríamos escribir un procedimiento para procesar algunos datos cuando se requiriese alterar los valores de esos datos.

De base a base

El programa ValorBase ilustra el uso de procedimientos con parámetros de valor. Se puede seleccionar cualquier base entre 2 (binaria) a 16 (hexadecimal), y los números decimales entrados desde el teclado se visualizarán en la notación apropiada. Por ejemplo, 32767 (MaxInt en varios compiladores pequeños de PASCAL) se convertiría en 7FFF en hexadecimal, 111111111111111 en binario o 77777 en octal.

Intente implementar estas modificaciones:

- Para manejar la representación negativa, debemos convertir el número a su complemento a dos. A nivel máquina, esto se realiza negándolo y sumándole uno. ¿Puede pensar alguna forma sencilla de hacer esto en PASCAL?
- ¿Le interesaría, tal vez, efectuar conversiones en dirección contraria? De ser así, podría utilizar el programa como modelo para tomar un número de cualquier base (entre 2 y 16) y producir el resultado en forma decimal. ¿Podría incorporar esto como un procedimiento en el programa anterior? Una pista: piense en los datos y en la forma de enlazarlos.

```
PROGRAM         ValorBase      (input,output);
        {convierte numeros decimales a CUALQUIER base en
        la escala de binaria a hexadecimal}
CONST
        Columnas  =  79; {para pantalla/impresora}

TYPE
        byte          = 0..255;
        cardinal      = 0..MaxInt;

VAR
        numero,
        base          : integer;
        Harto,
        legal         : boolean;

        {1111111111111111111111111111111}

PROCEDURE  EscribirDigito    (digito : byte);
     {valor de lista [contador] pasada a digito}
  BEGIN
    IF digito IN [0..9]
      THEN
        write (digito : 1)
      ELSE              {representar como A .. F}
        CASE digito OF
          10          : write ('A');
          11          : write ('B');
          12          : write ('C');
          13          : write ('D');
          14          : write ('E');
          15          : write ('F');
        END           {CASE}

  END;  {EscribirDigito}
        {1111111111111111111111111111111}

PROCEDURE  Imprimir      ( N            : cardinal;
                           base         : byte);
  CONST  {todos estos datos son locales a Imprimir}
         MaxDigitos=32;

  TYPE
         limites    = 1..MaxDigitos;
  VAR
         lista        : ARRAY [limites] OF byte;
         indice       : limites;
         contador     : byte;
  BEGIN
    contador := 0;

    REPEAT
         contador := succ (contador);
         lista [contador] := N MOD base;
         N := N DIV base

    UNTIL N=0;
                 {imprimir comenzando por BmsS:}
    FOR contador := contador DOWNTO 1 TO
      EscribirDigito (lista [contador])

  END;   {Imprimir}
         {1111111111111111111111111111111}
BEGIN    {ValorBase - Programa principal}

  REPEAT
    WriteLn ('Elegir una base numerica : ');
    WriteLn ('2... 16                       '):Columnas);
    WriteLn ('(con cualquier otra se sale)':Columnas);
    write ('Base?');
    read (base);
    legal := base IN [2 .. 16];

    IF legal THEN
      BEGIN
        write ('Numero ( 0 cambia base ) ? ' );
        read (numero) ;
        Harto := numero < = 0;

        WHILE NOT Harto DO
          BEGIN
            write (numero : Columnas DIV 2, ' a base ', base :
            1, 'es ');
            Imprimir ( numero, base );
            WriteLn;
            write ( 'Numero ? ');
            read (numero );
            Harto := numero<=0;
          END
      END

  UNTIL NOT legal

END.
```



Con el próximo fascículo se pondrán a la venta las tapas correspondientes al séptimo volumen.



El juego de tapas va acompañado de un sobre con los transferibles, numerados del 1 al 8, correspondientes a los volúmenes de que consta la obra; esto le permitirá marcar el lomo de cada uno de los volúmenes, a medida que aumente su colección.

Para encuadernar los 12 fascículos que componen un volumen, es preciso arrancar previamente las cubiertas de los mismos.

No olvide que, antes de colocar los fascículos en las tapas intercambiables, debe usted estampar el número en el lomo de las mismas, siguiendo las instrucciones que se dan a continuación:



1 Desprenda la hojita de protección y aplique el transferible en el lomo de la cubierta, haciendo coincidir los ángulos de referencia con los del recuadro del lomo.

2 Con un bolígrafo o un objeto de punta roma, repase varias veces el número, presionando como si quisiera borrarlo por completo.

3 Retire con cuidado y comprobará que el número ya está impreso en la cubierta. Cúbralo con la hojita de protección y repita la operación anterior con un objeto liso y redondeado, a fin de asegurar una perfecta y total adherencia.



Cada sobre de transferibles contiene una serie completa de números, del 1 al 8, para fijar a los lomos de los volúmenes. Ya que en cada volumen sólo aplicará el número correspondiente, puede utilizar los restantes para hacer una prueba preliminar.

Con el próximo número se ponen a la venta las tapas intercambiables para encuadernar 12 fascículos de

mi COMPUTER

Cada juego de tapas va acompañado de una colección de transferibles, para que usted mismo pueda colocar en cada lomo el número de tomo que corresponda



9 788485 822836
10083